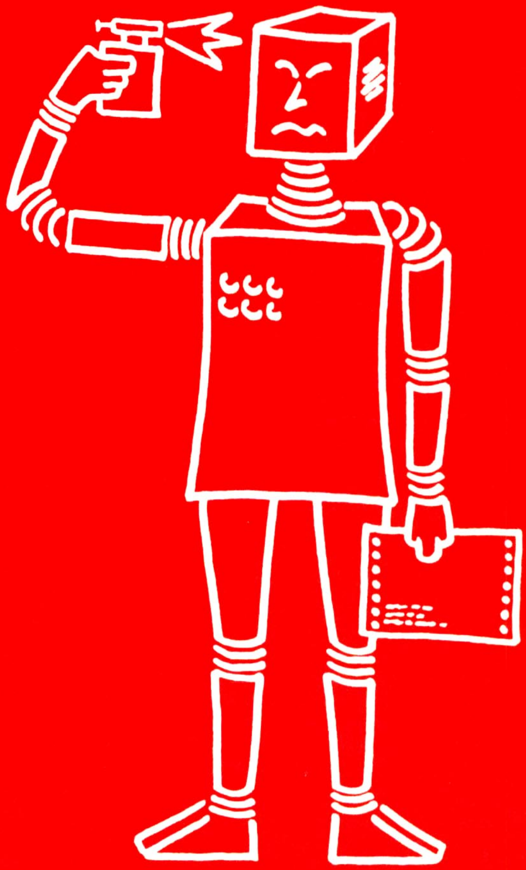
Universität Karlsruhe
Institut für Informatik I
Postfach 6380
D-7500 Karlsruhe
West Germany

Universität Kaiserslautern
Fachbereich Informatik
Postfach 3049
D-6750 Kaiserslautern
West Germany

SEKI MEMO

SEKI-PROJEKT



MECHANICAL GENERATION OF SORTS IN CLAUSE SETS

Manfred Schmidt-Schauss

MEMO SEKI-85-VI-KL

MECHANICAL GENERATION
OF SORTS IN CLAUSE SETS

Manfred Schmidt-Schauss

MEMO SEKI-85-VI-KL

Abstract

The algorithm SOGEN is described, which transforms a SIG-sorted clause set CS into a SIG′-sorted clause set CS′, where the output clause set is smaller, but the sort structure is more sophisticated.

This produced clause set is the input for our Theorem Prover, which has ΣRP* , an extension of ΣRP as its basic deductive calculus. Both calculi have resolution and paramodulation as their basic operations.

We prove that the transformation induced by SOGEN does not affect unsatisfiability, respectively satisfiability, of the clause set.

Introduction.

The advantages of a many-sorted calculus in automated reasoning systems are well known [Hay71, Hen72, Wa83, GM84, GM85, Co83, CD83, Ob62].: In a many-sorted calculus we obtain a shorter refutation of a smaller set of shorter clauses, as compared to the unsorted version.

To exploit the power of a many-sorted calculus, it is necessary that the problem to be solved has a sort structure and that it is presented in it's sorted version to the Theorem Prover. Usually this many-sorted input is hand-coded. There are examples, where the sort structure is naturally given, but there are also examples, for which this hand-coding is a hard task. Moreover this coding by hand may be faulty or not (un-)satisfiability preserving for some reasons.

In [Wa83,Sch85, Ob62] it is proved in the so called Sort-Theorem, that for special kinds of clause sets the transformation into a sorted version preserves unsatisfiability. But the direction of transformation described there is from the sorted version to the unsorted version (the relativization). However the input clause set is not of this form in general.

A further motivation for such an automatic transformation are the troubles in using a knowledge base with definitions and lemmas together with a sort-structure, since this requires a global (very unflexible) sort-structure. This limitation may be too strong and precludes the usage of sorts in such knowledge bases. But once a translation module is available, a knowledge base can be built up without sorts. The translator module preprocesses the input clause sets and prepares a sorted version for the Theorem Prover.

The question whether such a transformation does affect or not the (un-)satisfiability of clause sets needs a well-suited notion of the semantics of such a transformation. The right notion of a model in our case is the somewhat adapted notion of models in their original meaning, but not the Herbrand-models. The (adapted) E-model provides a very natural semantic for such transformations and shows how to design correct rules.

1) Basic Notions of a Many-Sorted Calculus.

We define the notions of signatures, sorts and algebras similar to those in the ΣRP^*-calculus [Sch85], but we drop some conditions on polymorphic functions. Such more general definitions are needed, since the rules of SOGEN, which we introduce in the next chapter, allow an "ad hoc" polymorphism of functions, which is not allowed in ΣRP^*.

1.1 Definition. (generalized signature)
A signature SIG is a triple $(\mathbb{S}, \mathbb{F}, \mathbb{P})$, where

i) $\mathbb{S}$ is the finite set of sorts, ordered by the reflexive and transitive relation $\le$ (possibly not antisymmetric). $\top$ is the greatest element of $\mathbb{S}$. (i.e. for all $S \in \mathbb{S}$: $S \le \top$). The ordering $\le$ is extended to tuples of sorts and means componentswise $\le$.

ii) $\mathbb{F}$ is the set of function symbols. $\mathbb{F} = \bigcup \mathbb{F}_W$, where $\mathbb{F}_W$ is the set of functions of arity n and signature $\emptyset \neq W \subseteq \mathbb{S}^{n+1}$. The sets $\mathbb{F}_W$ are pairwise disjoint.

If $\mathbb{F}_W \neq \emptyset$, then W satisfies:

1) The sort of constants is unique: i.e. $W \subseteq \mathbb{S}^1 \Rightarrow |W| = 1$.

2) If $W \subseteq \mathbb{S}^{n+1}$ for $n \ge 1$, W contains a greatest element $(S_{W,1},...,S_{W,n+1})$, i.e. for all $(S_1,...,S_{n+1}) \in W$: $(S_1,...,S_{n+1}) \le (S_{W,1},...,S_{W,n+1})$.

iii) $\mathbb{P}$ is the set of predicate symbols. $\mathbb{P}_D$ is the set of predicates of arity n with domain $D \in \mathbb{S}^n$. It is $\mathbb{P} = \bigcup \mathbb{P}_D$.

iv) For every sort $S \in \mathbb{S}$, there exists a constant c of sort $S_c \le S$, i.e. SIG is sensible in the sense of [HO80]. ∎

We use the following additional notation and abbreviations:

- $SO(f) = W$, iff $f \in \mathbb{F}_W$.
- $SO(P) = D$, iff $P \in \mathbb{P}_D$.
- $\mathbb{C}$ denotes the set of constants.
- $\mathbb{C}_S$ denotes the set of constants of sort S.
- $R \sqcap S = \{T \in \mathbb{S} \mid T \le R \text{ and } T \le S\}$
- $R \wedge S$ denotes the least element of $R \sqcap S$, provided this set is not empty and there is exactly one least element.

The following is the standard definition of a heterogeneous algebra (see e.g. [HO80]) with the additional proviso that the subsort relation is represented as the subset relation.

1.2 Definition. (algebra with respect to SIG.)
Let SIG be a signature. The pair (A,SIG) is an algebra of type SIG, iff the following conditions hold:

i) A is a nonempty set (the carrier).

ii) For every sort $S \in \mathbb{S}$, there is a related subset $S^A \subseteq A$, such that

1) $\top^A = A$

2) $\forall R,S \in \mathbb{S}: R \le S \Rightarrow R^A \subseteq S^A$.

iii) For every sort S and every constant $c \in \mathbb{C}_S$, there is a related $c^A \in S^A$.

iv) For every function $f \in \mathbb{F} \setminus \mathbb{C}$, there is a related function $f^A: A^n \to A$, such that for every $(S_1,...,S_{n+1}) \in SO(f)$ and every $a_i \in S_i^A$, $1 \le i \le n$: $f^A(a_1,...,a_n) \in S_{n+1}^A$. ∎

From the definition of a signature 1.1 iv), we have that $S^A \neq \emptyset$ for every sort S. Furthermore, if there are sorts R,S such that $S \le R$ and $R \le S$, then their representations are identical, i.e. $R^A = S^A$.

We extend the notion of a homomorphism to a $\mathbb{S}$-homomorphism, which respects the sort structure.

1.3 Definition. ($\mathbb{S}$-homomorphism of algebras)
Let SIG be a signature and let (A,SIG) and (B,SIG) be algebras of type SIG.
A mapping $\varphi: A \to B$ is a $\mathbb{S}$-homomorphism, iff
i) $\forall S \in \mathbb{S}: \varphi(S^A) \subseteq \varphi(S^B)$
ii) $\varphi f^A(a_1,...,a_n) = f^B(\varphi a_1,..., \varphi a_n)$ for all $f \in \mathbb{F}$ and all $a_i \in S_{f,i}{}^A$, $1 \le i \le n$, where $(S_{f,1},...S_{f,n+1})$ is the greatest element of SO(f). ■

Obviously, the composition of two $\mathbb{S}$-homomorphisms is again a $\mathbb{S}$-homomorphism.
Let $\mathbf{V}_S$ be the infinite set of variables of sort S, which we assume to be pairwise disjoint.
Let $\mathbf{V} = \bigcup \mathbf{V}_S$ be the set of all variables. Let $\mathbf{T}$ be the set of all (i.e. including ill-sorted) terms. That is, $\mathbf{T}$ is the least set with $\mathbf{V} \subseteq \mathbf{T}$, $\mathbb{C} \subseteq \mathbf{T}$ and $f(t_1,...,t_n) \in \mathbf{T}$ for all $f \in \mathbb{F}$ and all $t_i \in \mathbf{T}$.

We define the sort of a term t, namely GS(t), as a set of sorts. Intuitively, this is the set of sorts S, such that t could be substituted for a variable of sort S.

1.4 Definition.
Let SIG be a signature. Then the (generalized) sort of a term is defined by the mapping $GS: \mathbf{T} \to 2^{\mathbb{S}}$:

$$GS(t) = \begin{cases} \{R \mid R \ge S\}, & \text{if t is a variable or constant of sort S (i.e. } t \in \mathbf{V}_S \cup \mathbb{C}_S). \\ \{R \mid \text{there exist } S_i \in GS(t_i), 1 \le i \le n, \text{ and a } S_{n+1} \le R, \text{ such that } (S_1,...,S_{n+1}) \in SO(f)\} \\ & \text{if } t = f(t_1,...,t_n). \end{cases}$$

For example: In the sort structure of the complex numbers: GS(1) = {COMPLEX, REAL, INT, NAT }. The set GS(t) has the property, that $\forall S \in GS(t)$: $\{R \in \mathbb{S} \mid S \le R\} \subseteq GS(t)$.
We define the set $\mathbf{WST}$, the set of well-sorted terms as the set $\{t \in \mathbf{T} \mid GS(t) \neq \emptyset\}$.
Obviously for every term $t = f(t_1,...,t_n) \in \mathbf{WST}$, the subterms t_i are also in $\mathbf{WST}$.

1.5 Lemma. Let SIG be a signature. Then ($\mathbf{WST}$, SIG) is an algebra of type SIG, if the operations and the representations of sorts are defined as follows:
i) $f^{\mathbf{WST}}(t_1,...,t_n) = f(t_1,...,t_n)$
ii) $S^{\mathbf{WST}} = \{t \mid S \in GS(t)\}$.
Proof. We verify the conditions of definition 1.2:
i) Obviously $\mathbf{WST}$ is not empty.
ii) 1) $\top^{\mathbf{WST}} = \mathbf{WST}$, since $\top \in GS(t)$ for every well-sorted term t.
   2) Let $R,S \in \mathbb{S}$ and let $R \le S$. We have to show, that $R^{\mathbf{WST}} \subseteq S^{\mathbf{WST}}$.
   Let $t \in R^{\mathbf{WST}}$. Then $R \in GS(t)$. Since $R \le S$, $S \in GS(t)$. Hence $t \in S^{\mathbf{WST}}$.
iii) Let $c \in \mathbb{C}_S$ be a constant. Then $c^{\mathbf{WST}} = c$, $S \in GS(c)$. Hence $c^{\mathbf{WST}} \in S^{\mathbf{WST}}$.
iv) Let $t_i \in S_i^{\mathbf{WST}}$, $1 \le i \le n$, and let $(S_1,...,S_{n+1}) \in SO(f)$. The definition of $S_i^{\mathbf{WST}}$ gives $S_i \in GS(t_i)$, $1 \le i \le n$. Now the definition 1.4 yields $S_{n+1} \in GS(f(t_1,...,t_n))$.
   Hence $f(t_1,...,t_n) \in S_{n+1}{}^{\mathbf{WST}}$.

($\mathbf{WST}$,SIG) is the free algebra of type SIG. ($\mathbf{WST}_{gr}$,SIG) is the initial algebra of type SIG, where the suffix "gr" denotes ground objects (i.e. objects without variables). For proofs we refer to [Sch85].

1.6 Definition. A mapping $\sigma: \mathbf{WST} \to \mathbf{WST}$, which is identical almost everywhere is a $\mathbb{S}$-substitution, iff it is an endomorphism of the algebra $\mathbf{WST}$. ■

Let $\boldsymbol{\Sigma}$ denote the set of all $\mathbb{S}$-substitutions.

1.7 Lemma. Let SIG be a signature and let σ: $\mathbf{WST} \to \mathbf{WST}$ be a mapping. Then σ is a $\mathbb{S}$-substitution, iff the following conditions hold:

i) $\sigma c = c$, for all $c \in \mathbb{C}$.

ii) $\sigma f(t_1,...,t_n) = f(\sigma t_1,..., \sigma t_n)$ for all terms $f(t_1,...,t_n)$.

iii) $GS(x) \subseteq GS(\sigma x)$ for all variables x.

iv) $\{x \in \mathbb{V} \mid \sigma x \neq x\}$ is finite.

Proof.

"$\Rightarrow$" Let σ be a $\mathbb{S}$-substitution. Then σ is a $\mathbb{S}$-endomorphism. The only nontrivial condition is iii). Let $x \in \mathbb{V}$ and let $S \in GS(x)$. Then $x \in S^{\mathbf{WST}}$. 1.3 i) yields $\sigma x \in S^{\mathbf{WST}}$. Hence $S \in GS(\sigma x)$.

"$\Leftarrow$" Let i) - iv) be satisfied. We show only that $\sigma(S^{\mathbf{WST}}) \subseteq S^{\mathbf{WST}}$.

We use induction on the term structure.

Base case. For $x \in S^{\mathbf{WST}}$ we have $S \in GS(x)$, hence by iii) $S \in GS(\sigma x)$. This implies $\sigma x \in S^{\mathbf{WST}}$. For constants, trivially $c \in S^{\mathbf{WST}}$ implies $\sigma c \in S^{\mathbf{WST}}$.

Induction step. Let $t = f(t_1,...,t_n) \in \mathbf{WST}$. We show $GS(\sigma t) \supseteq GS(t)$. Let $R \in GS(t)$, then there exist $S_i \in GS(t_i)$, $1 \leq i \leq n$, and a $S_{n+1} \leq R$, such that $(S_1,...,S_{n+1}) \in SO(f)$. The induction hypothesis yields $S_i \in GS(\sigma t_i)$. From Lemma 1.5 and definition 1.2 iv) we conclude, that $f^{\mathbf{WST}}$ maps $(S_1{}^{\mathbf{WST}}, ..., S_n{}^{\mathbf{WST}})$ onto $S_{n+1}{}^{\mathbf{WST}}$.

Hence $f(\sigma t_1,..., \sigma t_n) = \sigma t \in S_{n+1}{}^{\mathbf{WST}}$, which implies $S_{n+1} \in GS(\sigma t)$.

Finally $S_{n+1} \leq R$ implies $R \in GS(\sigma t)$. ■

We shortly describe some needed notions:

$P(t_1,...,t_n)$ is an atom, where P is a predicate symbol and the t_i's are terms such that $S_i \in GS(t_i)$, where $SO(P) = (S_1,...,S_n)$. A (well-sorted) literal is a signed atom. The set of all well-sorted literals is called $\mathbb{L}$. A clause is a set of literals, i.e. an abbreviation for the disjunction of the literals, where all variables are universally quantified. A ground atom, a ground literal or a ground clause is one without variables. Instances of atoms, literals and clauses are their images under a $\mathbb{S}$-substitution. Equality ($\equiv$) is a distinguished binary predicate with domainsorts $SO(\equiv) = (\top,\top)$.

1.8 Definition. Let SIG be a signature. SIG is a polymorphic signature [Sch85], iff the following (additional) condition is satisfied:

i) $\langle \mathbb{S}, \leq \rangle$ is a partially ordered set.

ii) For every $f \in \mathbb{F}$, every $(S_1,...,S_{n+1}) \in SO(f)$ and every $(T_1,...T_n) \in \mathbb{S}^n$:

$$(T_1,...T_n) \leq (S_1,...,S_n) \Rightarrow \exists_1\, T_{n+1} \in \mathbb{S} : T_{n+1} \leq S_{n+1} \;\wedge\; (T_1,...T_{n+1}) \in SO(f). \blacksquare$$

The next lemma shows the connection between the sort in polymorphic signatures and the generalized sort in the signatures considered in this paper.

1.9 Lemma. Let SIG be a polymorphic signature. Then the following holds:

i) For all $t \in \mathbf{WST}$: $GS(t)$ contains a unique least element, which we denote with $[t]$.

ii) For all $t = f(t_1,...,t_n) \in \mathbf{WST}$: $([t_1],..., [t_n], [t]) \in SO(f)$.

Proof. We show i) and ii) by induction on the term structure of t.

Base case. For $t \in \mathbb{V}_S$ or $t \in \mathbb{C}_S$: $[t] = S$, and $GS(t) = \{R \in \mathbb{S} \mid R \geq S\}$. Since $\langle \mathbb{S}, \leq \rangle$ is a partial ordering, S is the unique least element of $GS(t)$.

Induction step. Let $t= f(t_1,...,t_n)$. Let $S_i = [t_i]$ $1 \leq i \leq n$. Let $R \in GS(t)$. Then there exist $R_i \in GS(t_i)$ and a $R_{n+1} \leq R$, such that $(R_1,...,R_n,R_{n+1}) \in SO(f)$. We have $(S_1,...,S_n) \leq (R_1,...,R_n)$. Hence (by definiton 1.8) there exists a unique $S_{n+1} \leq R_{n+1}$, such that $(S_1,...,S_{n+1}) \in SO(f)$. Now S_{n+1} is the unique least element of $GS(t)$. ■

Since a clause set CS is said to be satisfiable, if and only if a model for CS exists, it is necessary to give a precise definiton of a model with respect to a signature.

1.10 Definition. Let CS be a SIG-sorted clause set. An E-model for CS is a triple (D,SIG,R), such that the following conditions are satisfied:

i) (D,SIG) is an algebra of type SIG.

ii) For every predicate P there exists a relation $P^D \in R$ of the same arity.

iii) All clauses in CS are valid under all **S**-homomorphisms φ: **WST** $\rightarrow$ D. I.e. all clauses in CS are valid under all assignments of values in D to variables in clauses, where sorts are respected. (We say a literal $P(t_1,\dots,t_n)$ is valid under φ, iff $\varphi\ (P(t_1,\dots,t_n)) = P^D(\varphi t_1,\dots \varphi t_n)$ is valid i.e. $(\varphi t_1,\dots \varphi t_n)$ is in the relation P^D.)

iv) The equality predicate $\equiv$ is represented as the identity on D.

Remark. A Theorem of Herbrand states, that D could be chosen in such a way, that D is the image of $\mathbf{WST}_{gr}$ under all **S**-homomorphisms.

Furthermore, if no equality literals are in D, we can choose $D = \mathbf{WST}_{gr}$.

If all equality literals are unit-clauses, then we can choose $D = \mathbf{WST}_{gr}/\sim$, where $\sim$ is the congruence relation on $\mathbf{WST}_{gr}$, which is induced by all such unit-equalities.

2. The Algorithm SOGEN.

The goal of this chapter is to present the algorithm SOGEN, which transforms unary predicates into appropriate sorts, generating a polymorphic signature and a corresponding clause set from a given unsorted clause set. The algorithm is formulated in production rules. The correctness of each rule is shown in chapter 3.

2.1 Preliminaries for SOGEN.

The algorithm SOGEN needs a memory for already introduced relations on sorts and relationships between sorts and predicates to characterize the situation, where predicates and their corresponding sort are simultaneously present. It is not possible to express this in the signature. We call this set SC (sort constraints) and consider the members of SC as a special type of clauses. (The constraints in SC could be coded as special clauses; see rules RSC3 and RSC5) In the following we write P, if we mean a signed predicate. Now we specify SC:

SC is a set of pairs and triples:

1) A pair $(P,S_P) \in SC$, where $P \in \mathbb{P}$ and $S_P \in \mathbb{S}$ stands for "P is transformed into S_P". This means that for every term t, the sort of t is S_P, iff P(t) holds.
2) A triple (R,S,T) with $R,S,T \in \mathbb{S}$ represents $R \cap S = T$. This means that for every term t: if t is of sort S and of sort R, then t is of sort T. ■

Let $\leq$ be a reflexive, transitive relation on a finite set U. Then we denote with $MIN_\leq(U)$ a set, such that:

i) $MIN_\leq(U) \subseteq U$.
ii) $\forall u,v \in MIN_\leq(U)$: $u \leq v \Rightarrow u = v$.
iii) $\forall u \in U, \exists v \in MIN_\leq(U)$: $v \leq u$

Such a subset exists, since $\leq$ is reflexive and transitive.

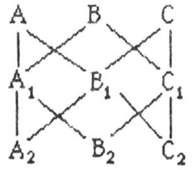
During the run of SOGEN, information about the intersections of sorts is available (sort constraints in the set SC). From this information, some new relations on sorts are deducable, for example, that two sorts are in the relation $\leq$ or that two sorts are "equal". For rules, which manipulate the set SC or which deduce this new information, we need some definitons.

2.1.1 Definition. Let SIG be a signature and let SC be a set of sort constraints. We define the set $REP_{SC}(S)$, which is the set of all sets $\{S_1,...,S_n\}$, which satisfy "$S_1 \cap ... \cap S_n = S$":

a) For every $S \in \mathbb{S}$, we define the set $REP_{SC}(S)$ of representations recursively:
   i) $\{S\} \in REP_{SC}(S)$.
   ii) If $\{S_1,...,S_j,...,S_n\} \in REP_{SC}(S)$ and $(R_1,R_2,S_j) \in SC$, then $MIN_\leq(\ (\{S_1,...,S_n\}\setminus\{S_j\}) \cup \{R_1,R_2\}\ ) \in REP_{SC}(S)$.
   iii) If $RP_1, RP_2 \in REP_{SC}(S)$, then $MIN_\leq(RP_1 \cup RP_2) \in REP_{SC}(S)$.
b) We define a relation $\leq_{SC}$ on sets of sorts, which is only used for elements of some $REP_{SC}(S)$:
   $\{S_1,...,S_n\} \leq_{SC} \{T_1,...,T_m\}$, iff for every $T \in \{T_1,...,T_m\}$, there exists an element $S \in \{S_1,...,S_n\}$, such that $S \leq T$.
   $\{S_1,...,S_n\} \leq_{SC} \{T_1,...,T_m\}$ can be interpreted as "$S_1 \cap ... \cap S_n \subseteq T_1 \cap ... \cap T_m$".
c) The base set for intersections is defined as:
   $BASE_{SC} = \{S \in \mathbb{S} \mid$ all elements of $REP_{SC}(S)$ are sets with exactly one element. $\}$. i.e. all sorts, which have only trivial representations.
d) Similarly $REP_{SC}(\{S_1,...,S_n\})$ is defined for sets of sorts. The intended meaning is to represent "$S_1 \cap ... \cap S_n$".
e) For $R,S \in \mathbb{S}$, we define $R \sim_{SC} S$, iff there exist $RP_{R1}, RP_{R2} \in REP_{SC}(R)$ and $RP_{S1}, RP_{S2} \in REP_{SC}(S)$ such that $RP_{R1} \leq_{SC} RP_{S1}$ and $RP_{S2} \leq_{SC} RP_{R2}$.

Two sorts R,S which satisfy $R \sim_{SC} S$ have always the same set as representation in an E-model, hence they can be identified in the sort structure.

2.1.2 Example. Let $A,B,C,A_1,B_1,C_1, A_2,B_2,C_2 \in \mathbb{S}$ and let (A,B,A_1), (A,C,B_1), (B,C,C_1), (A_1,B_1,A_2), (A_1,C_1,B_2), $(B_1,C_1,C_2) \in SC$. The following diagram shows the relationships:



From the set SC, we get the following relations:
$\{A,B\} \in REP_{SC}(A_1)$; $\{A,C\} \in REP_{SC}(B_1)$; $\{B,C\} \in REP_{SC}(C_1)$; $\{A,B,C\} \in REP_{SC}(A_2)$; $\{A,B,C\} \in REP_{SC}(B_2)$; $\{A,B,C\} \in REP_{SC}(C_2)$; Hence we have:
$A_2 \sim_{SC} B_2 \sim_{SC} C_2$, which reflects the fact, that $\cap$ is associative, commutative and idempotent. The following computation shows, what happened:
$A_2 = A_1 \cap B_1 = A \cap B \cap A \cap C = A \cap B \cap C = A \cap B \cap B \cap C = A_1 \cap C_1 = B_2$.

2.1.3 Example. Let A,B,C,D be sets, such that $A \cap B = C \cap D$ holds.
Then $B \cap C \supseteq A \cap B$, since $A \cap B = (A \cap B) \cap (C \cap D)$.
Without the rule 2.1.1 a) iii) this relation is not deducable with the representation mechanism.

2.1.4 Example. This example demonstrates, that in definition 2.3.1 e) in general $RP_{R1} \neq RP_{S1}$ and $RP_{R2} \neq RP_{S2}$:
Let A,B,C,D,E be sets and let $F = A \cap B = C \cap D \cap E$, $G = B \cap C = A \cap D$.
Then F is represented by {F},{A,B}, {C,D,E}, {A,B,C,D,E} and G is represented by {G}, {B,C}, {A,D}, {A,B,C,D}. (We use $F \leq A,B,C,D,E$ and $G \leq A,B,C,D$).
We have: $\{A,B,C,D\} \leq_{SC} \{A,B\}$ and $\{A,B,C,D,E\} \leq_{SC} \{A,B,C,D\}$. Hence $F \sim_{SC} G$. ∎

In the following we describe the rules of SOGEN by their input (IN) and their output (OUT), respectively by their condition and action.

2.2 Basic Transformation Rules.

Rule BT1. Introduction of sort

- IN
  - a) SIG
  - b) CS — CS contains a clause C, whose literals all have the same unary predicate P.
  - c) SC — There is no pair (P,S) for some S in SC.
- OUT
  - a) SIG′ — $\mathbb{S}' = \mathbb{S} \cup \{S_P\}$, S_P is a new sort symbol, c is a new constant of sort S_P. $S_P \leq S_{DP}$ is added, where $S_{DP} = SO(P)$. $\leq'$ is the transitive closure of $\leq$.
  - b) CS′ — CS
  - c) SC′ — $SC \cup \{(P,S_P)\}$.

Rule BT2. Changing sorts of constants.

- IN
  - a) SIG — $c \in \mathbb{C}_{S_c}$
  - b) CS — CS contains the clause {P(c)}
  - c) SC — SC contains (P,S_P) and a triple (S_P,S_c,T)
- OUT
  - a) SIG′ — $\mathbb{C}_{S_c}' = \mathbb{C}_{S_c} \setminus \{c\}$; $\mathbb{C}_T' = \mathbb{C}_T \cup \{c\}$
  - b) CS′ — CS
  - c) SC′ — SC

Rule BT3. Introdcution of sort relations.

IN a) SIG

b) CS CS contains the clause $\{P(x)\}$, where $[x] = S_x$.

c) SC $(P,S_P) \in SC$

OUT a) SIG′ $\mathbb{S}' = \mathbb{S}$, but $S_x \le S_P$ is added and $\le'$ is the transitive closure of $\le$.

b) CS′ CS

c) SC′ SC

Rule BT4. Changing the sort of a variable.

IN a) SIG

b) CS CS contains the clause $C = \{-P(x)\} \cup A$, where $[x] = S_x$.

c) SC $(P,S_P) \in SC$ and $(S_P,S_x,T) \in SC$.

OUT a) SIG′

b) CS′ $CS' = (CS \setminus \{C\}) \cup \{C'\}$, where $C' = A'$ and x is replaced by a new variable y of sort T.

c) SC′ SC

Rule BT5. Adding tuples to SO(f).

IN a) SIG

b) CS CS contains the clause $C = \{P(f(x_1,\ldots,x_n))\}$, where $[x_i] = S_i$ and the variables x_i are pairwise different.

c) SC $(P,S_P) \in SC$

OUT a) SIG′ $SO'(f) = SO(f) \cup \{(S_1,\ldots,S_n,S_P)\}$.

b) CS′ CS

c) SC′ SC

2.3 Deduction and Deletion Rules.

Rule DD1. Deductions.

IN a) SIG

b) CS

c) SC

OUT a) SIG′

b) CS′ $CS \cup \{C\}$, where C is a resolvent, factor or paramodulant of clauses in CS.

c) SC′ SC

Rule DD2. Clause Deletion Rules.

IN a) SIG

b) CS CS contains the clause C, which satisfies one of the following conditions:

i) C is subsumed by another clause C′ in C, i.e. there exists a substitution σ, such that $\sigma C' \subseteq C$.

ii) C is a pure clause, i.e. $C = \{L\} \cup A$, L is a literal, the predicate P of L is not the equality predicate, neither(P,S_1) nor $(-P,S_2)$ is in SC, and there exists no complementary literal in any of the clauses of CS.

iii) C is a tautology. i.e. either $C = \{L\} \cup \{-L\} \cup A$ or $C = \{P(t)\} \cup A$, $(P,S_P) \in SC$ and $S_P \in GS(t)$.

c) SC

OUT a) SIG′

b) CS′ $CS \setminus \{C\}$

c) SC′ SC

Rule DD3. Literal Deletion Rule (implicit resolution).

IN a) SIG
b) CS CS contains $C = \{-P(t)\} \cup A$, where $S_P \in GS(t)$
c) SC SC contains (P, S_P).

OUT a) SIG′
b) CS′ $(CS \setminus \{C\}) \cup \{A\}$.,
c) SC′ SC

2.4 Manipulations Based on SC.

Rule SC1. Trivial Intersection Properties.

IN a) SIG
b) CS
c) SC

OUT a) SIG′
b) CS′
c) SC′ $SC' = SC \cup \{(S_1,S_2,S_2) \mid S_1 \geq S_2\} \cup \{(S_2,S_1,T) \mid (S_1,S_2,T) \in SC\}$.

Rule SC2. Introduction of sort relations by representations.

IN a) SIG There exist $S,T \in \mathbb{S}$ and $RP_S \in REP_{SC}(S)$ and $RP_T \in REP_{SC}(T)$ such that $RP_S \leq_{SC} RP_T$ and not $S \leq T$.
b) CS
c) SC

OUT a) SIG′ $S \leq T$ is added to $\leq$. $\leq'$ is the transitive closure of $\leq$.
b) CS′
c) SC′

Rule SC3. Application of contraposition.

IN a) SIG contains $S_1 \leq S_Q$.
b) CS
c) SC contains the pairs (P,S_P), $(-P,S_{-P})$, (Q,S_Q), $(-Q,S_{-Q})$, and the triples (S,S_P,S_1), (S,S_{-Q},S_2).

OUT a) SIG′ $S_2 \leq S_{-P}$ is added to $\leq$. $\leq'$ is the transitive closure of $\leq$.
b) CS′
c) SC′

Rule SC4. Introducing the intersection of two sorts.

IN a) SIG $S_1, S_2 \in \mathbb{S}$ and $S_1 \cap S_2 \neq \emptyset$.
b) CS
c) SC does not contain (S_1,S_2,S) nor (S_2,S_1,S).

OUT a) SIG′ $\mathbb{S}' = \mathbb{S} \cup \{S_N\}$, S_N is a new sort with $S_N \leq' S_1$, $S_N \leq' S_2$, and $S \leq' S_N$ for all $S \in S_1 \cap S_2$. $\leq'$ is the transitive closure of $\leq$.
b) CS′
c) SC′ $SC' = SC \cup \{(S_1,S_2,S_N)\}$.

2.5 Manipulating the sort structure itself.

Rule MS1. Deletion of cycles in $\langle \mathbb{S}, \leq \rangle$.

IN a) SIG There exist sorts $S, T \in \mathbb{S}$, such that $S \neq T$, $S \leq T$ and $T \leq S$.

b) CS

c) SC

OUT a) SIG′ $\langle \mathbb{S}', \leq' \rangle = \langle \mathbb{S}/\sim , \leq/\sim \rangle$, where $\sim$ is defined as: $T \sim S$, iff $T \leq S$ and $T \geq S$.
In SO′(f) and SO′(P) sorts are replaced by their equivalence class.

b) CS′ CS, where all sorts are replaced by their equivalence class.

c) SC′ SC, where all sorts are replaced by their equivalence class.

2.6 Manipulations of the signature.

Rule SO1. Making f a polymorphic funtion.

IN a) SIG $\langle \mathbb{S}, \leq \rangle$ is cycle free. $(S_{f,1},\ldots,S_{f,n+1})$ is the greatest element of SO(f). The following condition is satisfied:
For every $(S_1,\ldots,S_{n+1})$, $(T_1,\ldots,T_{n+1}) \in SO(f)$:

$$(\forall i = 1,\ldots,n \;\; S_i \sqcap T_i \neq \emptyset) \Rightarrow ((\forall i = 1,\ldots,n \;\; S_i \wedge T_i \text{ is unique}) \text{ and there exists a sort } R_{n+1} \text{, such that } S_{n+1} \geq R_{n+1},\; T_{n+1} \geq R_{n+1} \text{ and } (S_1 \wedge T_1,\ldots,S_n \wedge T_n, R_{n+1}) \in SO(f).)$$

b) CS

c) SC

OUT a) SIG′ where SO′(f) =

$$\left\{ (S_1,\ldots S_{n+1}) \;\middle|\; \begin{array}{l} S_i \leq S_{f,i}\text{, for } i = 1,\ldots,n \text{ and} \\ S_{n+1} \text{ is the least element of the set} \\ \{S \mid (S_1',\ldots S_n', S) \in SO(f) \text{ and } S_i \leq S_i' \} \end{array} \right\}$$

b) CS′ CS

c) SC′ SC

Rule SO2. Adding intersections of range-sorts.

IN a) SIG $(S_1,\ldots,S_{n+1})$, $(T_1,\ldots,T_{n+1}) \in SO(f)$ and $S_i \sqcap T_i \neq \emptyset$ for $i = 1,\ldots,n$
$S_{n+1} \sqcap T_{n+1} = \emptyset$.

b) CS

c) SC

OUT a) SIG′ $\mathbb{S}' = \mathbb{S} \cup \{S_N\}$, where S_N is a new sort. c is a new constant of sort S_N.
$S_N \leq S_{n+1}$ and $S_N \leq T_{n+1}$ is added. $\leq'$ is the transitive closure of $\leq$.

b) CS′ CS

c) SC′ SC

Rule SO3 Adding a tuple of intersection sorts.

IN a) SIG $(S_1,\ldots,S_{n+1})$, $(S_1',\ldots,S_{n+1}') \in SO(f)$ and $(T_1,\ldots,T_{n+1}) \notin SO(f)$

b) CS

c) SC $(S_i, S_i', T_i) \in SC$ for $i = 1,\ldots,n+1$

OUT a) SIG′ $SO'(f) = SO(f) \cup \{(T_1,\ldots,T_{n+1})\}$.

b) CS′ CS

c) SC′ SC

Rule SO4 SO(f)-Restriction.

IN a) SIG $f \in \mathbb{F}$

b) CS a term or subterm (in CS) starting with f exists.

c) SC

OUT a) SIG′ $SO'(f) = \{(S_1,...,S_{n+1}) \in SO(f) \mid (S_1,...,S_{n+1}) \le (T_1,...,T_{n+1})\}$., where
$(T_1,...,T_{n+1})$ is an appropriate tuple, such that
$(T_1,...,T_{n+1}) \le (S_{f,1},...,S_{f,n+1})$ and all literals in CS′ remain well-sorted.
b) CS′ CS
c) SC′ SC

Rule SO5 SO(P)-Restriction.
IN a) SIG $P \in \mathbb{P}$
b) CS a literal starting with predicate P is in CS.
c) SC does not contain (P,S_P) or $(-P,S_{-P})$.
OUT a) SIG′ SO′(P) is changed into $(S_1,...,S_n) \le SO(P)$, such that all literals in CS′ remain well-sorted.
b) CS′ CS
c) SC′ SC

Remark. If SIG is a polymorphic signature in the rules SO4 and SO5 and $\langle S,\le \rangle$ is a semilattice, then the changes for SO(f) and SO(P) are uniquely determined.

Rule SO6 Deleting functions and constants from the signature.
IN a) SIG $f \in \mathbb{F}$. ($c \in \mathbb{C}$)
b) CS f does not occur in a literal of CS. (c does not occur in a literal of CS.)
c) SC
OUT a) SIG′ f is removed from SIG. (c is removed from SIG.)
b) CS′ CS
c) SC′ SC

2.7 Reducing SC.

Rule RSC1 Trivial cases.
IN a) SIG contains $T \ge S$
b) CS
c) SC contains (T,S,S) or (S,T,S)
OUT a) SIG′
b) CS′ CS
c) SC′ SC \ {(T,S,S), (S,T,S)}.

Rule RSC2 non complementary predicates.
IN a) SIG
b) CS neither P nor -P occurs in CS.
c) SC contains (P,S_P), but no pair $(-P,S_{-P})$
OUT a) SIG′ P is removed from SIG.
b) CS′ CS
c) SC′ SC \ $\{(P,S_P)\}$.

Rule RSC3 complementary predicates. (general case).
IN a) SIG
b) CS neither P nor -P occurs in CS.
c) SC contains (P,S_P) and $(-P,S_{-P})$
OUT a) SIG′ P is removed from SIG. Two new functions f_+ and f_- are added to $\mathbb{F}$.
With $SO(P) = S_{DP}$, the (not polymorphic) functions have S_{DP} as their domain and S_P and S_{-P} as their range respectively.
b) CS′ $CS \cup \{(\forall x{:}S_P, y{:}S_{-P}\ x \ne y)\} \cup \{(\forall x{:}S_{DP}, x = f_+(x) \vee x = f_-(x))\}$.
The last clause is the skolemized form of :
$\{\forall x{:}S_{DP}, (\exists z{:}S_P\ x = z) \vee (\exists z{:}S_{-P}\ x = z)\}$.

c) SC′ $SC \setminus \{(P,S_P)\ (-P,S_{-P})\}$.

Remark: The functions f_+ and f_- are in fact skolem functions.

Rule RSC4 complementary predicates. (a special case).

IN a) SIG $P \in \mathbb{P}$, $SO(P) = S_{DP}$. For every ground term t:

$S_{DP} \in GS(t) \Rightarrow S_P \in GS(t) \vee S_{-P} \in GS(t)$

b) CS neither P nor -P occurs in CS. CS contains an equality literal

c) SC contains (P,S_P) and $(-P,S_{-P})$

OUT a) SIG′ P is removed from SIG.

b) CS′ $CS \cup \{(\forall x{:}S_P,\ y{:}S_{-P}\ \ x \neq y)\}$

c) SC′ $SC \setminus \{(P,S_P),\ (-P,S_{-P})\}$.

Rule RSC5 complementary predicates. (a special case).

IN a) SIG $P \in \mathbb{P}$, $SO(P) = S_{DP}$. For every ground term t:

$S_{DP} \in GS(t) \Rightarrow (S_P \in GS(t) \Longleftrightarrow\ S_{-P} \notin GS(t)\ )$.

b) CS neither P nor -P occurs in CS. CS contains no equality literal

c) SC contains (P,S_P) and $(-P,S_{-P})$

OUT a) SIG′ P is removed from SIG.

b) CS′ CS

c) SC′ $SC \setminus \{(P,S_P),\ (-P,S_{-P})\}$.

Rule RSC6 Removing intersection information (general case).

IN a) SIG

b) CS

c) SC contains (S_1,S_2,T), where $S_1 \neq T$, $S_2 \neq T$ and $S_1 \wedge S_2 = T$

OUT a) SIG′ a new (skolem) function g is added to SIG, where g has domain-sort S_1 and range-sort T and $(S,S) \in SO(g)$ for all $S \leq T$

b) CS′ $CS \cup \{(\forall x{:}S_1,\ y{:}S_2\ ,\ x \neq y \vee g(x) = x)\}$

(The new clause is the optimized and skolemized form of $\forall x{:}S_1,\ y{:}S_2\ \ x = y\ \Rightarrow (\exists z{:}T\ \ x = z\ )\ )$

c) SC′ $SC \setminus \{(S_1,S_2,T),(S_2,S_1,T)\}$.

Rule RSC7 Removing intersection information (a special case).

IN a) SIG

b) CS $=$ occurs only in unit-clauses. For every triple (S,T,S') and for every literal $s = t$, which follows semantically ($\models$) from the equality clauses in CS, where $S \in GS(s)$ and $T \in GS(t)$ hold, there exists a term $t_{S'}$, such that $S' \in GS(t_{S'})$ and $s = t_{S'}$ follows semantically from the equality clauses in CS.

c) SC For every triple $(S_1,S_2,S_3) \in SC$: $S_1 \wedge S_2 = S_3$

OUT a) SIG′

b) CS′

c) SC′ SC \ {all triples in SC}.

Rule RSC8 Removing intersection information (a special case).

IN a) SIG

b) CS there is no equality literal in CS.

c) SC

OUT a) SIG′

b) CS′

c) SC′ SC \ {all triples in SC}.

2.8 Analysis by cases.

Rule AC1. Adding the tautology $((\forall x\ -P(x)) \vee (\exists y\ P(y)))$

IN a) SIG $SO(P) = S_{DP}$.
b) CS contains P
c) SC does not contain (P,S_P)

OUT i) a) SIG′ $\mathbb{C}' = \mathbb{C} \cup \{c\}$, where c is a new constant of sort S_{DP}.
b) CS′ $CS \cup \{P(c)\}$
c) SC′ SC

OUT ii) a) SIG′
b) CS′ $CS \cup \{\forall x{:}S_{DP}\ -P(x)\}$
c) SC′ SC

Rule AC2. For constants c either P(c) or -P(c).

IN a) SIG $SO(P) = S_{DP}$. c is a constant of sort $S \leq S_{DP}$, $S \nleq S_P$, $S \nleq S_{-P}$
b) CS
c) SC contains (P,S_P) and $(-P,S_{-P})$

OUT i) a) SIG′
b) CS′ $CS \cup \{\{P(c)\}\}$
c) SC′ SC

OUT ii) a) SIG′
b) CS′ $CS \cup \{\{-P(c)\}\}$
c) SC′ SC

Rule AC3. Using $\{(\forall x{:}S\ P(x)) \vee (\forall x{:}S\ -P(x)) \vee ((\exists y{:}S\ P(y)) \wedge (\exists z{:}S\ -P(z))\}$

IN a) SIG $SO(P) = S_{DP}$, $S \in \mathbb{S}$ and $S \leq S_{DP}$, $S \nleq S_P$, $S \nleq S_{-P}$
b) CS
c) SC contains (P,S_P) and $(-P,S_{-P})$

OUT i) a) SIG′ SIG
b) CS′ $CS \cup \{\{\forall x{:}S\ P(x)\}\}$
c) SC′ SC

OUT ii) a) SIG′ SIG
b) CS′ $CS \cup \{\{\forall x{:}S\ -P(x)\}\}$
c) SC′ SC

OUT iii) a) SIG′ c_+, c_- are new constants of sort S.
b) CS′ $CS \cup \{\{P(c_+)\}\} \cup \{\{-P(c_-)\}\}$
c) SC′ SC

Rule AC4. Splitting a clause into two clauses.

IN a) SIG $SO(P) = S_{DP}$
b) CS CS contains a clause C, such that there exists an $x \in \mathbb{V}(C)$, with $[x] = S \leq S_{DP}$.
c) SC contains (P,S_P), $(-P,S_{-P})$, (S,S_P,S_1), (S,S_{-P},S_2).

OUT a) SIG′ SIG
b) CS′ $CS \setminus \{C\} \cup \{C_1, C_2\}$, where C_i is the clause C, but the variable x is replaced by $x_i{:}\ S_i$.
c) SC′ SC

2.9 Termination Conditions.

CO1 If SC contains (P,S_P) and $(-P,S_{-P})$ and there exists a $S \in \mathbb{S}$, such that $S \leq S_P$ and $S \leq S_{-P}$, then the signature contains a contradiction.

CO2 If the clause set is empty and for all (P,S_P), $(-P,S_{-P}) \in SC$, where $SO(P) = S_{DP}$ and for all $S \in \mathbb{S}$: $S \le S_{DP} \Rightarrow (S \le S_P \Leftrightarrow S \nleq S_{-P})$ then the original clause set is satisfiable.

CO3 If some clause is empty, then a refutation has been found.

CO4 If no rule besides the rules RSCi is applicable, but some clause contains a literal $\pm P(t)$ and a pair (P,S_P) or $(-P,S_{-P})$ is in SC, then the algorithm SOGEN failed.

2.10 Manipulations Caused by Equalities.

Rule EQ1. Existence of an intersection sort.

IN a) SIG
b) CS CS contains a clause $\{s \equiv t\}$, $S \in GS(s)$, $T \in GS(t)$ and $S \cap T = \emptyset$
c) SC

OUT a) SIG′ $\mathbb{S}' = \mathbb{S} \cup \{S_N\}$, S_N is a new sort symbol; c is a new constant of sort S_N. $S_N \le S$ and $S_N \le T$ is added. . $\le'$ is the transitive closure of $\le$.
b) CS′ CS
c) SC′ $SC \cup \{(S,T,S_N)\}$.

Rule EQ2. The sort of a constant is changed.

IN a) SIG contains a constant c of sort S_c.
b) CS CS contains a clause $\{c \equiv t\}$, $S_t \in GS(t)$.
c) SC contains (S_c,S_t,S_{ct}).

OUT a) SIG′ the sort of c is changed into S_{ct}.
b) CS′ CS
c) SC′ SC

Rule EQ3. New Sort Relations.

IN a) SIG
b) CS CS contains a clause $\{x \equiv t\}$, $T \in GS(t)$. and x is a variable of sort S.
c) SC

OUT a) SIG′ $S \le T$ is added. $\le'$ is the transitive closure of $\le$.
b) CS′ CS
c) SC′ SC

Rule EQ4. New tuples in SO(f).

IN a) SIG $(S_1,\ldots,S_{n+1}) \in SO(f)$
b) CS CS contains a clause $\{f(x_1,\ldots,x_n) \equiv t\}$, $T \in GS(t)$ and the x_i are distinct variables of sort S_i.
c) SC contains (S_{n+1},T,S')

OUT a) SIG′ $SO'(f) = SO(f) \cup \{(S_1,\ldots,S_n,S')\}$.
b) CS′ CS
c) SC′ SC

2.11 A Rule for Unary Functions.

Rule UC1. Introducing a new predicate.

IN a) SIG $SO(P) = S_{DP}$.
b) CS CS contains a literal $\pm P(g(t))$
c) SC

OUT a) SIG′ P_g is a new predicate with $SO(P_g) = (\top)$.

b) CS' $CS'' \cup \{(\forall x_i{:}S_i\ -P(g(x_i)) \vee P_g(x_i))\} \cup \{(\forall x_i{:}S_i\ P(g(x_i)) \vee -P_g(x_i))\}$ for all i,
where $\{S_1,\ldots,S_n\} = MAX_{\leq}\{T_1 \mid (T_1,T_2) \in SO(g), T_2 \leq S_{DP}\}$.
CS'' is the clause set CS, where all literals of the form $\pm P(g(t))$ are replaced by the $\pm P_g(t)$.

c) SC' SC

Remark. The clauses $\{(\forall x_i{:}S_i\ -P(g(x_i)) \vee P_g(x_i))\}$ and $\{(\forall x_i{:}S_i\ P(g(x_i)) \vee -P_g(x_i))\}$ give rise to a tuple for the function g (with rule BT5).

2.12 How the Rules Work.

We describe, which rules are tightly connected and which combination of rules solve some subproblems, such as making the signature polymorphic. Furthermore we give the sequence in which the blocks of rules should be applied and say, which rule to apply first.

The priority of the rules is essential, since the set of rules without any priority may run in a loop.

1) The rules BT2, BT3, BT4, BT5, DD2 and DD3 have highest priority. They should be applied, whenever possible. Every application of a rule BT2, BT3, BT4, BT5 could be followed by the deletion of the corresponding literal.
2) The rule BT1 should be applied, whenever possible, but the restriction is given, that it may be possible, that the transformation of one or more unary predicates is inhibited since a control module knows, that the transformation of this sorts is not possible or incomplete.
3) The rules SC1, SC2, SC4, MS1 form a block of rules, which is able to complete the sort structure, such that for all sorts S_1,S_2 either $S_1 \wedge S_2$ exists or $S_1 \sqcap S_2 \neq \emptyset$. In this block, the rule SC4 must have lowest priority, since the uncontrolled application of SC3 alone does not terminate.
4) The rule SC3 makes it possible to code more information into the signature. It avoids, that the relations between the sorts S_P depend on the sequence of application of the rules. For example the clause $P \rightarrow Q$ is equivalent to $-Q \rightarrow -P$, but if the relation $S_P \leq S_Q$ is generated, then the clause is deleted, but the relation $S_{-Q} \leq S_{-P}$ may be missing.
5) The rules SO1, SO2, SO3 together with the rules of 3) i.e. SC1, SC2, SC4, MS1 are able to make the signature polymorphic. The priority of rules should be: SO2, SO3, SC1, SC2, MS1, SC4, SO1.
6) The rule SO4 and SO6 may be used to delete unnecessary information from SO(f) (resp. the signature). This reduces the set of well-sorted terms, and possibly the conditions for the rule RSC5 are satisfied after application of this rules.
7) The rules RSCi should fire, if no other rules are applicable. In practical applications, the addition of clauses by the rules RSC4 and RSC6 is very unpleasant, since they introduce equality literals. They may be used to indicate, that the transformation is possibly incomplete.
8) The rules ACi need some control, since it depends on global information or knowledge, which of this applications may contribute to a proof or not. Note that every application of a rule ACi could be followed by a rule BTi.
9) The rules EQi are not essential, (we have not implemented these rules) since on the one hand, an equality reasoning module exploits unit-equalities much better, and on the other hand, in the connection graph calculus, all unifiers in all links have to be recomputed after application of these rules.
10) The rule UC1 is relevant only for a decision procedure for the corresponding clause set.

3. Soundness and Completeness of SOGEN.

In this chapter we show, that all rules preserve satisfiability repectively inconsistency of the combination clause set + sort constraints + signature. Therefore a notion of satisfiability (inconsistency) is needed, which is given in a preliminary paragraph.
For a certain set of rules, we show that they terminate. Furthermore we prove, that SOGEN provides a decision algorithm for clause sets, where all predicates and functions are unary.

3.1 Some Preliminary Definitions and Lemmas.

3.1.1 Definition. Notion of a model for CS and SC.
Let SIG = $(\mathbb{S}, \mathbb{F}, \mathbb{P})$ be a signature. Let CS be a clause set and SC be a set of sort constraints.

We say the CS + SC have an E-model (D,SIG,R), iff

i) (D,SIG,R) is an E-model for CS.
ii) For all $(P,S_P) \in SC$: $S_P^D = \{d \mid d \in S^D$ and $P^D(d)$ is valid$\}$, where $SO(P) = (S)$
iii) For all $(R,S,T) \in SC$: $R^D \cap S^D = T^D$.

In the sequel we deal with signatures SIG and SIG'. We sometimes abbreviate $\mathbf{WST}_{SIG}$ with $\mathbf{WST}$ and $\mathbf{WST}_{SIG'}$ with $\mathbf{WST}'$. The same holds for the symbols like SO,$\mathbb{S}$,$\mathbb{F}$, and $\mathbb{P}$.
We say a rule is sound, iff it preserves the satisfiability of CS + SC, a rule is said to be complete, iff it preserves the inconsistency of CS + SC. We sometimes cite lemmas, which are proved only for the case SC = $\emptyset$. But all proofs are adaptable to the case SC $\neq \emptyset$ in a straigth forward way, since all clauses in SC could be coded as clauses (see rules RSC4 and RSC5).
The next definiton provides the notion of embedded algebras, which frequently occurs in the rules of SOGEN.

3.1.2 Definition. (embedding of two algebras.)
Let SIG = $(\mathbb{S}, \mathbb{F}, \mathbb{P})$ and SIG' = $(\mathbb{S}', \mathbb{F}', \mathbb{P}')$ be signatures and let (D,SIG) and (D',SIG') be algebras of type SIG, respectively SIG'.
We say (D,SIG) is embedded in (D',SIG'), iff the following conditions are satisfied:

i) $\mathbb{S} \subseteq \mathbb{S}'$, $\mathbb{F} \subseteq \mathbb{F}'$.
ii) $\mathbf{WST} \subseteq \mathbf{WST}'$
iii) D = D', and D, D' have the same representations for $S \in \mathbb{S}$ and $f \in \mathbb{F}$.

3.1.3 Lemma. Lifting and restriction of $\mathbb{S}$-homomorphisms in embedded algebras.
Let SIG = $(\mathbb{S}, \mathbb{F}, \mathbb{P})$ and SIG' = $(\mathbb{S}', \mathbb{F}', \mathbb{P}')$ be signatures and let (D,SIG) be an algebra, which is embedded in (D',SIG').

Then we have:

i) For every $\mathbb{S}$-homomorphism φ: $\mathbf{WST} \to D$, there exists an $\mathbb{S}$-homomorphism φ': $\mathbf{WST}' \to D$, such that $\varphi'_{|\mathbf{WST}} = \varphi$.
ii) For every $\mathbb{S}$-homomorphism φ': $\mathbf{WST}' \to D$, the restriction $\varphi'_{|\mathbf{WST}}$ is an $\mathbb{S}$-homomorphism $\varphi'_{|\mathbf{WST}}$: $\mathbf{WST} \to D$

Proof.

i) We define $\varphi'(x) = \varphi(x)$ for all $x \in \mathbb{V}$. Since $\mathbf{WST}'$ is a free algebra with respect to SIG' and since condition 3.1.2 iii) is satisfied, this defines uniquely a $\mathbb{S}$-homomorphism φ': $\mathbf{WST}' \to D$, since $c^D = c^{D'}$ and $f^D = f^{D'}$. Obviously, $\varphi'_{|\mathbf{WST}} = \varphi$.
ii) From 3.1.2 iii) we conclude that the restriction $\varphi'_{|\mathbf{WST}}$ is an $\mathbb{S}$-homomorphism with respect to SIG. ∎

The next lemma provides a tool for proving that a rule of SOGEN is sound and complete.

3.1.4 Lemma. Let SIG,SIG′ be signatures, such that $\mathbb{L} \subseteq \mathbb{L}'$. Let CS be a clause set, where all literals are in $\mathbb{L}$. Let SC be a set of sort constraints for SIG.

Then the following holds:

i) Given an E-model (D,SIG,R) for CS + SC and an algebra (D′,SIG′), such that (D,SIG) is embedded in the algebra (D′,SIG′), then there exists an E-model (D,SIG′,R′) for CS + SC.

ii) Given an E-model (D,SIG′,R′) for CS + SC and an algebra (D,SIG), such that (D,SIG) is embedded in the algebra (D′,SIG′), then there exists an E-model (D,SIG,R) for CS + SC.

Proof.

We can assume, that the equality predicate is in $\mathbb{P}$.

i) Let (D,SIG,R) be an E-model for CS + SC and let (D′,SIG′) be an algebra, such that (D,SIG) is embedded in the algebra (D′,SIG′).
We define R′ = R, and show, that (D,SIG,R′) is an E-model for CS + SC with respect to SIG′. The conditions 3.1.1 ii) and iii) are trivially satisfied, since the representations of sorts and predicates are not changed. The equality is still represented as the identity.
Now let $\varphi' : \mathbf{WST} \to D$ be an $\mathbb{S}$-homomorphism with respect to SIG′.
Then by lemma 3.1.3 ii) $\varphi = \varphi'|_{\mathbf{WST}}$ is an $\mathbb{S}$-homomorphism with respect to SIG.
Since all literals in clauses of CS are in $\mathbb{L}$, the images of clauses under φ and φ' are the same. Thus $\varphi'C$ is valid for all $C \in CS$, since $\varphi'C = \varphi C$ and (D,SIG,R) is a E-model.

ii) Let (D,SIG′,R′) be an E-model for CS + SC and let (D,SIG) be an algebra , such that (D,SIG) is embedded in the algebra (D′,SIG′).
We define R = R′, and show that (D,SIG,R) is an E-model for CS + SC:
We show the nontrivial part:
Let $\varphi : \mathbf{WST} \to D$ be an $\mathbb{S}$-homomorphism with respect to SIG. Then by lemma 3.1.3 i) there exists an $\mathbb{S}$-homomorphism $\varphi' : \mathbf{WST}' \to D$ (with respect to SIG), such that $\varphi'|_{\mathbf{WST}} = \varphi$. Since all literals in clauses of CS are in $\mathbb{L}$, the images of clauses under φ and φ' are the same. Thus $\varphi'C$ is valid for all $C \in CS$, since $\varphi'C = \varphi C$ and (D,SIG′,R) is an E-model. ■

The next lemma gives sufficient conditions, such that $\mathbf{WST} \subseteq \mathbf{WST}'$, which is one of the basic preconditions for an algebra D to be embedded in an algebra D′.

3.1.5 Lemma. Let SIG,SIG′ be signatures and let $\psi: \mathbb{S} \to \mathbb{S}'$ be a mapping.
Let the following conditions be satisfied:

i) $\mathbb{F} \subseteq \mathbb{F}'$

ii) $\psi\mathrm{T} = \mathrm{T}'$

iii) $\forall R,S \in \mathbb{S}: R \leq S \Rightarrow \psi R \leq' \psi S$

iv) For all $S \in \mathbb{S}$: $\mathbf{V}_S \subseteq \mathbf{V}_{\psi S}$

v) For $c \in \mathbb{C}_R$: c is contained in $\mathbb{C}_S'$, such that $S \leq' \psi R$

vi) For every $f \in \mathbb{F} \setminus \mathbb{C}$ and for every $(S_1,...,S_{n+1}) \in SO(f)$, there exists a tuple $(S_1',...,S_{n+1}') \in SO'(f)$ with $\psi(S_1,...,S_n) \leq' (S_1',...,S_n')$ and $S_{n+1}' \leq' \psi S_{n+1}$.

vii) For every $P \in \mathbb{P}$: $\psi SO(P) = SO(P')$

Then i) $\mathbf{WST} \subseteq \mathbf{WST}'$.

ii) $\mathbb{L} \subseteq \mathbb{L}'$.

Proof. We prove $\psi GS(t) \subseteq GS'(t)$ for all $t \in \mathbf{WST}$ by structural induction.

i) For all $t \in \mathbf{WST}$, we have $GS'(t) \supseteq \psi(GS(t)) \neq \emptyset$, hence $t \in \mathbf{WST}'$.

ii) Let $P(t_1,...,t_n)$ be a well-sorted literal. Then $S_i \in GS(t_i)$, where $SO(P) = (S_1,...,S_n)$.
Since $\psi S_i \in \psi GS(t_i) \subseteq GS'(t_i)$, this literal is also well-sorted with respect to SIG′. ■

Proof of $\psi GS(t) \subseteq GS'(t)$:

Base case. For $x \in \mathbb{V}$ we have $\psi GS(x) \subseteq GS'(x)$ by condition iv).

For $c \in \mathbb{C}_S$, we conclude from condition v), that $\psi GS(c) \subseteq GS'(c)$

Induction step. Let $t = f(t_1,...,t_n) \in \mathbb{WST}$ and let $S \in GS(t)$.

Then there exist $S_i \in GS(t_i)$, $i=1,...,n$ and $S_{n+1} \in \mathbb{S}$ such that $(S_1,...,S_{n+1}) \in SO(f)$ and $S_{n+1} \leq S$. Now the induction hypothesis implies $\psi S_i \in GS(t_i)$. Condition vi) yields, that there exists a $(S_1',...,S_{n+1}') \in SO'(f)$ with $\psi S_i \leq' S_i'$ and $S_{n+1}' \leq' \psi S_{n+1}$. This implies $S_{n+1}' \in GS'(t)$. From $\psi S \geq' \psi S_{n+1} \geq' S_{n+1}'$, we conclude $\psi S \in GS'(t)$. ■

Remark. The mapping ψ in the lemma above is usually the identity on $\mathbb{S}$, or the canonical mapping from $\mathbb{S}$ onto $\mathbb{S}/\sim$.

3.2 Soundness and Completeness of the Rules in 2.1

3.2.1 Lemma. The introduction of new sorts is sound and complete.

Rule BT1:

IN a) SIG

b) CS CS contains a clause C, whose literals have all the same unary predicate P

c) SC There is no pair (P,S) for some S in SC.

OUT a) SIG' $\mathbb{S}' = \mathbb{S} \cup \{S_P\}$, S_P is a new sort symbol, c is a new constant of sort S_P. $S_P \leq S_{DP}$ is added, where $S_{DP} = SO(P)$. $\leq'$ is the transitive closure of $\leq$.

b) CS' CS

c) SC' $SC \cup \{(P,S_P)\}$.

Proof. We show, that IN has an E-model, iff OUT has an E-model.

IN → OUT: Let (D,SIG,R) be an E-model of IN. We show, that the conditions of 3.1.4 i) are satisfied. Therefore we construct (D,SIG') as an extension of (D,SIG). Let $S_P^D = \{d \mid d \in S_{DP}^D$ and $P^D(d)$ is valid$\}$. We have $S_P^D \subseteq S_{DP}^D$.

Since IN has an E-model, there exists some $d_P \in D$, such that $P^D(d_P)$ is valid. We have $S_{DP}^D \neq \emptyset$. The conditions of Lemma 3.1.5 are satisfied, if we choose ψ as the identity on $\mathbb{S}$. Then $\mathbb{WST} \subseteq \mathbb{WST}'$, and $\mathbb{L} \subseteq \mathbb{L}'$. We take (D,SIG') as the algebra with the same representation as (D,SIG) on D, S_P^D as above and $c^D = d_P$. Now (D,SIG) is embedded in (D,SIG') and 3.1.4 i) is applicable.

OUT → IN. Let (D,SIG',R') be an E-model for OUT. We define (D,SIG) as the restriction of (D,SIG'). Then obviously (D,SIG) is embedded in (D,SIG') and 3.1.4 ii) is applicable. ■

3.2.2 Lemma. Changing the sort of a constant is sound and complete.

Rule BT2:

IN a) SIG $c \in \mathbb{C}_{Sc}$

b) CS CS contains the clause $\{P(c)\}$

c) SC SC contains (P,S_P) and a triple (S_P,S_c,T)

OUT a) SIG' $\mathbb{C}_{Sc}' = \mathbb{C}_{Sc} \setminus \{c\}$; $\mathbb{C}_T' = \mathbb{C}_T \cup \{c\}$

b) CS' CS

c) SC' SC

Proof.

IN → OUT: Let (D,SIG,R) be an E-model for IN. We show, that the algebra (D,SIG) can be considered as an algebra of type SIG'. It suffices to show, that $c^D \in T^D$. We have $c^D \in S_P^D$, since $(P,S_P) \in SC$ and obviously $c^D \in S_c^D$. This together with $(S_P,S_c,T) \in SC$ implies, that $c^D \in T^D = S_P^D \cap S_c^D$. The conditions of Lemma 3.1.5 are satisfied, if we choose ψ as the identity on $\mathbb{S}$, since $T \leq' S_c$. Now Lemma 3.1.4 i) gives an E-model for OUT.

OUT → IN: Let (D,SIG′,R′) be an E-model for OUT. Since $T \le S_C$, (D,SIG) is an algebra of type SIG. The conditions of Lemma 3.1.5 are satisfied, if we choose ψ as the identity on $\mathbb{S}$. Now Lemma 3.1.4 ii) gives an E-model for IN. ■

3.2.3 Lemma. The introduction of sort relations is sound and complete.

Rule BT3:

IN a) SIG
b) CS CS contains the clause $\{P(x)\}$, where $[x] = S_X$.
c) SC $(P,S_P) \in SC$

OUT a) SIG′ $\mathbb{S}' = \mathbb{S}$, but $S_X \le S_P$ is added and $\le'$ is the transitive closure of $\le$.
b) CS′ CS
c) SC′ SC

Proof.

IN → OUT: Let (D,SIG,R) be an E-model for IN. We show, that the algebra (D,SIG) can be considered as an algebra of type SIG′. Therefore it suffices to show, that $S_X{}^D \subseteq S_P{}^D$.
The conditions of Lemma 3.1.5 are satisfied, if we choose ψ as the identity on $\mathbb{S}$, hence $\mathbf{WST} \subseteq \mathbf{WST}'$. Let $d \in S_X{}^D$. There exists an $\mathbb{S}$-homomorphism φ: $\mathbf{WST} \to \mathbf{WST}'$, such that $\varphi x = d$. $\varphi\{P(x)\} = P^D(d)$ is valid, since (D,SIG,R) is an E-model, hence $d \in S_P{}^D$.
Now Lemma 3.1.4 i) gives an E-model for OUT.

OUT → IN: Let (D,SIG′,R′) be an E-model for OUT. Trivially, (D,SIG) is an algebra of type SIG. The conditions of Lemma 3.1.5 are satisfied, if we choose ψ as the identity on $\mathbb{S}$. Then Lemma 3.1.4 ii) gives an E-model for IN. ■

3.2.4 Lemma. Changing the sort of a variable is sound and complete.

Rule BT4:

IN a) SIG
b) CS CS contains the clause $C = \{-P(x)\} \cup A$, where $[x] = S_X$.
c) SC $(P,S_P) \in SC$ and $(S_P,S_X,T) \in SC$.

OUT a) SIG′
b) CS′ $CS' = (CS \setminus \{C\}) \cup \{C'\}$, where $C' = A'$ and x is replaced by a new variable y of sort T.
c) SC′ SC

Proof.

IN → OUT: Let (D,SIG,R) be an E-model for IN. We show, that (D,SIG,R) is an E-model for OUT. It suffices to show, that the changed clause is valid under all $\mathbb{S}$-homomorphisms. Let φ: $\mathbf{WST} \to$ D be an $\mathbb{S}$-homomorphism. From $T \le S_P$ and $T \le S_X$ we conclude, that $\varphi y \in S_P{}^D$ and $\varphi y \in S_X{}^D$. Since $\mathbf{WST}$ is free, there exists an $\mathbb{S}$-homomorphism π: $\mathbf{WST} \to$ D, such that $\pi x = \varphi y$ and $\pi|_{\mathbf{V}(A)\setminus\{x\}} = \varphi|_{\mathbf{V}(A)\setminus\{x\}}$. $\pi(\{-P(x)\} \cup A)$ is valid, since (D,SIG,R) is an E-model, but $\pi(\{-P(x)\})$ is not valid, because $\pi x \in S_P{}^D$. Hence $\pi(A)$ must be valid. $\pi(A) = \varphi(A')$ implies, that the new clause A′ is valid under φ.

OUT → IN: Let (D,SIG,R) be an E-model for OUT. Let φ: $\mathbf{WST} \to$ D be an $\mathbb{S}$-homomorphism. We determine, whether $\varphi(\{-P(x)\} \cup A)$ is valid or not.

CASE $\varphi x \notin S_P{}^D$.

Then $\varphi(\{-P(x)\})$ is true, hence C is valid under φ.

CASE $\varphi x \in S_P{}^D$.

The triple (S_X,S_P,T) is in SC, hence $\varphi x \in T^D$. Since $\mathbf{WST}$ is free, there exists an $\mathbb{S}$-homomorphism π: $\mathbf{WST} \to$ D, such that $\pi y = \varphi x$ and $\pi|_{\mathbf{V}(A)\setminus\{x\}} = \varphi|_{\mathbf{V}(A)\setminus\{x\}}$. $\pi(A')$ is valid in the E-model (D,SIG,R). $\pi(A') = \varphi(A)$ implies, that A′ is valid under φ. ■

3.2.5 Lemma. Adding tuples to SO(f) is sound and complete.

Rule BT5:

IN a) SIG

b) CS CS contains the clause $C = \{P(f(x_1,...,x_n))\}$, where $[x_i] = S_i$ and the variables x_i are pairwise different.

c) SC $(P,S_P) \in SC$

OUT a) SIG′ $SO'(f) = SO(f) \cup \{(S_1,...,S_n,S_P)\}$.

b) CS′ CS

c) SC′ SC

Proof.

IN → OUT: Let (D,SIG,R) be an E-model for IN. We show, that (D,SIG) can be considered as an algebra of type SIG′. Let $d_i \in S_i^D$, $i = 1,...,n$. We argue, that $f^D(d_1,...,d_n) \in S_P^D$. Since **WST** is free, there exists an **S**-homomorphism φ: **WST** → D, such that $\varphi x_i = d_i$, $i = 1,...,n$.

(D,SIG,R) is an E-model, hence $\varphi P(f(x_1,...,x_n)) = P^D(f^D(d_1,...,d_n))$ is valid. Now $f^D(d_1,...,d_n) \in S_p^D$, since $(P,S_P) \in SC$. The rest follows with Lemma 3.1.5 and 3.1.4 i).

OUT → IN: Let (D,SIG′,R′) be an E-model for OUT. Obviously (D,SIG) is embedded in (D,SIG′). Then Lemma 3.1.4 ii) is applicable. ■

3.3 Soundness and Completeness of the Rules DDi.

In this paragraph we give only proofs for the nonstandard deletion rules such as tautology deletion and a special kind of replacement resolution.

3.3.1 Lemma. The deletion of a tautology clauses $\{P(t)\} \cup A$, where $(P,S_P) \in SC$ and $S \in GS(t)$, is sound and complete.
(Rule DD2 iii) 2nd case)

Rule DD2. Clause Deletion Rules.

IN a) SIG

b) CS CS contains the clause C, which satisfies the following condition: $C = \{P(t)\} \cup A$, $(P,S_P) \in SC$ and $S_P \in GS(t)$. I.e. C is a tautology.

c) SC

OUT a) SIG′

b) CS′ $CS \setminus \{C\}$

c) SC′ SC

Proof. IN → OUT: trivial

OUT → IN: Let (D,SIG,R) be an E-model for OUT. For every **S**-homomorphism φ: **WST** → D: $\varphi t \in S_P^D$, hence $P^D(\varphi t)$ is valid. Thus the whole clause $\{P(t)\} \cup A$ is valid under φ . ■

3.3.2 Lemma. The rule DD3 (replacement resolution) is sound and complete.

Rule DD3:

IN a) SIG

b) CS CS contains $C = \{-P(t)\} \cup A$, where $S_P \in GS(t)$

c) SC SC contains (P,S_P).

OUT a) SIG′

b) CS′ $(CS\setminus \{C\}) \cup \{A\}$.,

c) SC′ SC

Proof.

IN → OUT: Let (D,SIG,R) be an E-model for IN. We show, that (D,SIG,R) is an E-model for OUT. Let φ: **WST** → D be an **S**-homomorphism. Then φC is valid. We have $\varphi t \in S_P^D$ and hence $P^D(\varphi t)$ is true. This means $\varphi(-P(t))$ is false. Thus φA is valid.

OUT → IN: trivial. ■

3.4 Soundness and Completeness of SC-Manipulations.

In this paragraph we prove, that the rules SCi are sound and complete. The first two lemmas show, that the representations REP_{SC}, defined in paragraph 2.1, have the intended meaning.

3.4.1 Lemma. Let (D,SIG,R) be an E-model for CS and SC.
Then: $\{S_1,...,S_n\} \in REP_{SC}(S) \Rightarrow S_1^D \cap ... \cap S_n^D = S^D$.

Proof. We verify the construction of the set REP_{SC}; see Definition 2.1.1. I.e. the proof is by induction.

i) $S \in REP_{SC}(S)$; Obviously $S^D = S^D$.

ii) Let $\{S_1,...,S_j,...,S_n\} \in REP_{SC}(S)$ and let $(R_1,R_2,S_j) \in SC$. We have $R_1^D \cap R_2^D = S_j^D$ and $S_1^D \cap ... \cap S_j^D \cap ... \cap S_n^D = S^D$ by the induction hypothesis. The replacement of S_j^D does not change the right side of the equation. Furthermore, if T_0^D, T_1^D are among the sets to be intersected, and $T_0 \leq T_1$, then $T_0^D \subseteq T_1^D$, and T_1^D can be removed.

iii) Similar (trivial) arguments show, that case iii) is also correct. ■

3.4.2 Lemma. Let $\{S_1,...,S_n\}$ and $\{T_1,...,T_m\}$ be sets of sorts such that $\{S_1,...,S_n\} \leq_{SC} \{T_1,...,T_m\}$. Then in every algebra representation, which corresponds to an E-model we have: $S_1^D \cap ... \cap S_n^D \subseteq T_1^D \cap ... \cap T_m^D$.

Proof. We show $S_1^D \cap ... \cap S_n^D \cap T_1^D \cap ... \cap T_m^D = S_1^D \cap ... \cap S_n^D$:
By the definition of $\leq_{SC}$, for every T_i there exists a S_j, such that $S_j \leq T_i$. That means $S_j^D \cap T_i^D = S_j^D$. Hence we can add successively the T_i^D to the right side of $S_1^D \cap ... \cap S_n^D = S_1^D \cap ... \cap S_n^D$, getting the desired equality. ■

3.4.3 Lemma. Adding trivial tuples to SC is sound and complete.

Rule RSC1 Trivial cases.

IN a) SIG contains $T \geq S$
b) CS
c) SC contains (T,S,S) or (S,T,S)

OUT a) SIG'
b) CS' CS
c) SC' SC \ {(T,S,S), (S,T,S)}.

Proof.
IN → OUT. Let (D,SIG,R) be a E-model of IN. For $S_1,S_2 \in \mathbb{S}$, $S_1 \geq S_2$ implies, that $S_1^D \supseteq S_2^D$, hence $S_1^D \cap S_2^D = S_2^D$.
OUT → IN. trivial. ■

3.4.4 Lemma. The introduction of sort relations by intersection representations is sound and complete.

Rule SC2:

IN a) SIG There exist $S,T \in \mathbb{S}$ and $RP_S \in REP_{SC}(S)$ and $RP_T \in REP_{SC}(T)$ such that $RP_S \leq_{SC} RP_T$ and not $S \leq T$.
b) CS
c) SC

OUT a) SIG' $S \leq T$ is added to $\leq$. $\leq'$ is the transitive closure of $\leq$.
b) CS' CS
c) SC' SC

Proof.
IN → OUT. Let (D,SIG,R) be an E-model of IN. Lemma 3.4.2 implies, that $S^D \subseteq T^D$. Thus (D,SIG) can be considered as an algebra of type SIG′. The rest follows with Lemma 3.1.5 and 3.1.4 i) in a standard way.
OUT → IN. trivial. ■

3.4.5 Lemma. The application of contraposition is sound and complete.
Rule SC3:
IN a) SIG contains $S_1 \le S_Q$.
b) CS
c) SC contains the pairs (P,S_P), $(-P,S_{-P})$, (Q,S_Q), $(-Q,S_{-Q})$; and the triples (S,S_P,S_1), (S,S_{-Q},S_2).
OUT a) SIG′ $S_2 \le S_{-P}$ is added to $\le$. $\le'$ is the transitive closure of $\le$.
b) CS′ CS
c) SC′ SC

Proof.
We apply some rules of the algorithm SOGEN, which we have proved to be sound and complete. It is allowed to use all rules in two directions.
i) From $S_1 \le S_Q$ we can introduce the tautology $\forall x{:}S_1\ Q(x)$.(Rule DD2) ($(Q,S_Q) \in SC$ and $S_Q \in GS(x)$, since $S_1 \le S_Q$.)
ii) We replace this clause by the clause $\forall x{:}S\ P(x) \Rightarrow Q(x)$.(Rule BT4) ($(S,S_P,S_1) \in SC$ and $(P,S_P) \in SC$).
iii) This is the same as $\forall x{:}S\ -Q(x) \Rightarrow -P(x)$. Then application of the rule BT4 yields the clause $\forall x{:}S_2\ -P(x)$. ($(S,S_{-Q},S_2) \in SC$ and $(-Q,S_{-Q}) \in SC$)
iv) Rule BT3 yields the relation $S_2 \le S_{-P}$. ■

3.4.6 Lemma. Adding the intersection of two sorts is sound and complete.
Rule SC4:
IN a) SIG $S_1, S_2 \in \mathbb{S}$ and $S_1 \sqcap S_2 \neq \emptyset$.
b) CS
c) SC does not contain (S_1,S_2,S) nor (S_2,S_1,S).
OUT a) SIG′ $\mathbb{S}' = \mathbb{S} \cup \{S_N\}$, S_N is a new sort with $S_N \le' S_1$, $S_N \le' S_2$, and $S \le' S_N$ for all $S \in S_1 \sqcap S_2$. $\le'$ is the transitive closure of $\le$.
b) CS′ CS
c) SC′ $SC' = SC \cup \{(S_1,S_2,S_N)\}$.

Proof.
IN → OUT. Let (D,SIG,R) be an E-model of IN. SIG′ is a signature, since SIG′ is strict. We construct an algebra (D,SIG′): Let $S_N{}^D = S_1{}^D \cap S_2{}^D$. Then all relations between sorts and their representing subsets of D satisfy Definition 1.2 ii). (D,SIG,R) is an E-model for SC′. Now by Lemma 3.1.5 and 3.1.4 i) there exists an E-model for CS′ and SC′.
OUT → IN. Follows trivially from Lemmas 3.1.5 and 3.1.4 ii) ■

3.4.7 Lemma. Making $\langle\mathbb{S},\le\rangle$ cycle free is sound and complete.
Rule MS1. Deletion of cycles in $\langle\mathbb{S},\le\rangle$.
IN a) SIG There exist sorts $S,T \in \mathbb{S}$, such that $S \neq T$, $S \le T$ and $T \le S$.
b) CS
c) SC
OUT a) SIG′ $\langle\mathbb{S}',\le'\rangle = \langle\mathbb{S}/\sim, \le'/\sim\rangle$, where $\sim$ is defined as: $T \sim S$, iff $T \le S$ and $T \ge S$. In SO′(f) and SO′(P) sorts are replaced by their equivalence class.
b) CS′ CS, where all sorts are replaced by their equivalence class.
c) SC′ SC, where all sorts are replaced by their equivalence class.

Proof.

IN → OUT. Let (D,SIG,R) be an E-model if IN. The relation ~ is an equivalence relation. Let $\psi: \mathbb{S} \to \mathbb{S}/\sim$ be the canonical mapping. For $S_1 \sim S_2$ we have $S_1^D = S_2^D$. Lemma 3.1.5 and Definition 3.1.2 together imply, that (D,SIG,R) is embedded in an algebra (D,SIG′,R). An E-model of OUT can be derived from Lemma 3.1.4 i).

OUT → IN. Let (D,SIG′,R) be an E-model if OUT. The same arguments as above yield an E-model of IN.

3.5 The Manipulation of SO is Sound and Complete. (Rules SOi)

3.5.1 Lemma. To make f polymorphic is sound and complete. (Rule SO1)

Rule SO1. Making f a polymorphic funtion.

IN a) SIG $\langle \mathbb{S}, \le \rangle$ is cycle free. $(S_{f,1},\dots,S_{f,n+1})$ is the greatest element of SO(f). The following condition is satisfied:

For every $(S_1,\dots,S_{n+1})$, $(T_1,\dots,T_{n+1}) \in SO(f)$:

$(\forall i = 1,\dots,n \ \ S_i \sqcap T_i \neq \emptyset) \Rightarrow ((\forall i = 1,\dots,n \ \ S_i \wedge T_i$ is unique) and there exists a sort R_{n+1}, such that $S_{n+1} \ge R_{n+1}$, $T_{n+1} \ge R_{n+1}$ and $(S_1 \wedge T_1,\dots,S_n \wedge T_n, R_{n+1}) \in SO(f)$.)

b) CS

c) SC

OUT a) SIG′ where SO′(f) =

$$\left\{ (S_1,\dots S_{n+1}) \;\middle|\; \begin{array}{l} S_i \le S_{f,i}\text{, for } i = 1,\dots,n \text{ and} \\ S_{n+1} \text{ is the least element of the set:} \\ \{ S \mid (S_1',\dots S_n', S) \in SO(f) \text{ and } S_i \le S_i' \} \end{array} \right\}$$

b) CS′ CS

c) SC′ SC

Proof.

IN → OUT. Let (D,SIG,R) be an E-model of IN.

i) We show, that the definition of SO′(f) makes sense.:
Let S_i, $1 \le i \le n$, be fixed, $S_i \le S_{f,i}$. Let MSS = $\{S \mid (S_1,\dots,S_n,S) \in SO(f), S_i \le S_i', 1 \le i \le n\}$.
Assume, that MSS contains two distinct minimal elements MS_1 and MS_2. Let $(T_1,\dots,T_n,MS_1)$ and $(T_1',\dots,T_n',MS_2)$ be (existing) (n+1) - tuples in SO(f) with $T_i \ge S_i$ and $T_i' \ge S_i$. The condition of Rule SO1 implies, (obviously $T_i \sqcap T_i' \neq \emptyset$) that there exists a sort MS_3 such that $MS_1 \ge MS_3$ and $MS_2 \ge MS_3$ and $(T_1 \wedge T_1',\dots, T_n \wedge T_n', MS_3) \in SO(f)$. Since $\langle \mathbb{S}, \le \rangle$ is cycle free and MS_1 and MS_2 are minimal, $MS_1 = MS_2 = MS_3$.

ii) We show, that the defined SO′(f) satisfies the conditions for a polymorphic function.:
Let $(S_1,\dots,S_{n+1}) \in SO'(f)$ and let $(T_1,\dots,T_n) \in \mathbb{S}^n$, such that $(T_1,\dots,T_n) \le (S_1,\dots,S_n)$. If a T_{n+1} exists, such that $(T_1,\dots,T_{n+1}) \in SO(f)$, then T_{n+1} is unique (by the definition of SO′(f)). In order to show, that such a T_{n+1} exists, is suffices to show, that the set MSS above is not empty. But the maximal range-sort of f is always in MSS.

iii) (D,SIG) can be considered as an algebra of type SIG′.
Let $d_i \in S_i^D$, $1 \le i \le n$ and let $(S_1',\dots,S_{n+1}') \in SO'(f)$,. Then $f^D(d_1,\dots,d_n) \in S^D$ for all $S \in \{S \mid (S_1,\dots,S_n,S) \in SO(f), S_i \le S_i', 1 \le i \le n\}$. Thus $f^D(d_1,\dots,d_n) \in S_{n+1}{}^D$.

iv) Now Lemma 3.1.5 can be applied in both directions, where ψ is the identity, since the condition 3.1.5 vi) is satisfied with $S_i = S_i'$, $1 \le i \le n$ for the direction $\mathbb{S} \to \mathbb{S}'$, and in the other direction $\mathbb{S}' \to \mathbb{S}$, the tuple in SO(f), which has the minimal element as range-sort is the desired one. Hence **WST = WST′** and (D,SIG) is embedded in (D,SIG'). Now Lemma 3.1.4 i) shows, that an E-model of OUT exists.

OUT → IN. We show only, that (D,SIG) is embedded in an algebra (D,SIG'), the other arguments are the same as above. Let $(S_1,...,S_{n+1}) \in SO(f)$ and let $d_i \in S_i^D$, $1 \leq i \leq n$. By the definition of SO'(f), there exists a $S_{n+1}' \leq S_{n+1}$ with $(S_1,..., S_n, S_{n+1}') \in SO'(f)$. Hence $f^D(d_1,...,d_n) \in S_{n+1}'^D \subseteq S_{n+1}^D$. ■

3.5.2 Lemma. The introduction of intersections of range sorts is sound and complete.

Rule SO2:

IN a) SIG $(S_1,...,S_{n+1}), (T_1,...,T_{n+1}) \in SO(f)$ and $S_i \sqcap T_i \neq \emptyset$ for $i = 1,...,n$

$S_{n+1} \sqcap T_{n+1} = \emptyset$.

b) CS

c) SC

OUT a) SIG' $\mathbb{S}' = \mathbb{S} \cup \{S_N\}$, where S_N is a new sort. c is a new constant of sort S_N.

$S_N \leq S_{n+1}$ and $S_N \leq T_{n+1}$ is added. ≤' is the transitive closure of ≤.

b) CS' CS

c) SC' SC

Proof.

IN → OUT: Let (D,SIG,R) be an E-model for IN. We construct (D,SIG'), such that (D,SIG) is embedded in (D,SIG'): Let $S_N^D = S_{n+1}^D \cap T_{n+1}^D$. There exist $d_i \in S_i^D \cap T_i^D$, $1 \leq i \leq n$. Then $f^D(d_1,...,d_n) \in S_N^D$. We define $c^D = f^D(d_1,...,d_n)$. By Lemma 3.1.5 we have **WST** ⊆ **WST'** and (D,SIG) is embedded in (D,SIG'). The rest follows with Lemma 3.1.4 i).

OUT → IN Let (D,SIG',R) be an E-model for OUT. Obviously (D,SIG) is embedded in (D,SIG'). The rest follows with Lemma 3.1.4.ii). ■

3.5.3 Lemma. Adding a tuple of intersection sorts is sound and complete.

Rule SO3:

IN a) SIG $(S_1,...,S_{n+1}), (S_1',...,S_{n+1}') \in SO(f)$ and $(T_1,...,T_{n+1}) \notin SO(f)$

b) CS

c) SC $(S_i,S_i',T_i) \in SC$ for $i = 1,...,n+1$

OUT a) SIG' $SO'(f) = SO(f) \cup \{(T_1,...,T_{n+1})\}$.

b) CS' CS

c) SC' SC

Proof.

IN → OUT: Let (D,SIG,R) be an E-model for IN. We show, that (D,SIG) is embedded in (D,SIG'): Let $d_i \in T_i^D$, $1 \leq i \leq n$. Then $d_i \in S_i^D$ and $d_i \in S_i'^D$, $1 \leq i \leq n$. Hence $f^D(d_1,...,d_n) \in S_{n+1}^D \cap S_{n+1}'^D = T_{n+1}^D$. By Lemma 3.1.5 we have **WST** ⊆ **WST'**. The rest follows with Lemma 3.1.4 i).

OUT → IN . trivial ■

3.5.4 Lemma. SO(f)-restriction is sound and complete. (Rule SO4)

Proof. Follows immediately from Lemma 3.1.5 and 3.1.4. ■

3.5.5 Lemma. SO(P)-restriction is sound and complete. (Rule SO5)

Proof. Follows immediately from Lemma 3.1.5 and 3.1.4. ■

3.6 Reducing SC.

In this paragraph it is proved, that the reformulation of conditions, which stem from SC, is sound and complete, and that under certain preconditions, these (undesired) clauses are not needed.

3.6.1 Lemma. Rule RSC1 is sound and complete.

Proof. trivial.

3.6.2 Lemma. Deleting (P,S_P) is sound and complete, if S_{-P} is not generated.

Rule RSC2:

IN a) SIG
b) CS neither P nor -P occurs in CS.
c) SC contains (P,S_P), but no pair $(-P,S_{-P})$

OUT a) SIG′ P is removed from SIG.
b) CS′ CS
c) SC′ $SC' = SC \setminus \{(P,S_P)\}$.

Proof.

IN → OUT: trivial.

OUT → IN. Let (D,SIG′,R′) be an E-model for OUT. We change the relation P^D of R′ in the following way: $P^D(d)$ should be valid ,iff $d \in S_P{}^D$. Then the resulting (D,SIG,R) is an E-model for IN, since the predicate P does not occur in clauses of CS and the constraint defined by $(P,S_P) \in SC$ is satisfied. ∎

3.6.3 Lemma. Deleting (P,S_P) and $(-P,S_{-P})$ from SC and adding the appropriate clause is sound and complete.

Rule RSC3.

IN a) SIG
b) CS neither P nor -P occurs in CS.
c) SC contains (P,S_P) and $(-P,S_{-P})$

OUT a) SIG′ P is removed from SIG. Two new functions f_+ and f_- are added to $\mathbb{F}$. With $SO(P) = S_{DP}$, the (not polymorphic) functions have S_{DP} as their domain and S_P and S_{-P} as their range respectively.
b) CS′ $CS \cup \{\{\forall x{:}S_P, y{:}S_{-P}\ x \neq y\}\} \cup \{\{\forall x{:}S_{DP}, x \equiv f_+(x) \vee x \equiv f_-(x)\}\}$.
c) SC′ $SC' = SC \setminus \{(P,S_P)\ (-P,S_{-P})\ \}$.

Proof.

IN → OUT: Let (D,SIG,R) be an E-model for IN.

We define an E-model for OUT. Let the algebra (D,SIG′) have the same representation as (D,SIG). We have to define the representation of f_+ and f_-: Let $d_{-P} \in S_{-P}{}^D$ and let $d_P \in S_P{}^D$ be fixed. $f_+(d) := d$, if $d \in S_P$ and $f_+(d) := d_P$, if $d \in S_{-P}$. $f_-(d) := d$, if $d \in S_{-P}$ and $f_-(d) := d_{-P}$, if $d \in S_P$.We define R′ to be R where the relation P^D is removed. Our task is to show, that (D,SIG′,R′) is an E-model for OUT. Let φ: **WST** → D be an **S**-homomorphism.

It suffices to show, that the new clauses are valid in the model.

$\varphi(x \neq y)$ is valid: Assume, that $\varphi(x \neq y)$ is not valid. Then $\varphi x = \varphi y = d$, where $d \in S_P{}^D \cap S_{-P}{}^D$. But this is impossible, since either $P^D(d)$ is valid or $-P^D(d)$ is valid (equivalently $P^D(d)$ is not valid) in (D,SIG,R).

$\varphi(\ f_+(x) \equiv x \vee f_-(x) \equiv x\ )$ is valid: Let $\varphi x = d$; then $d \in S_{DP}{}^D$ and either $P^D(d)$ or $-P^D(d)$ is valid, hence either $d \in S_P{}^D$ or $d \in S_{-P}{}^D$. Thus either $d = f_+(d)$ or $d = f_-(d)$. This means, that $\varphi(\ f_+(x) \equiv x \vee f_-(x) \equiv x\ )$ is valid.

OUT → IN: Let (D,SIG′,R′) be an E-model for OUT. We define an E-model (D,SIG,R) for IN. Let $R = R' \cup \{P^D\}$, and let $P^D(d)$ be valid, iff $d \in S_P$. It suffices to show, that the constraints (P,S_P) and (P,S_{-P}) are satisifed in (D,SIG,R). We have $\{d \mid -P^D(d) \text{ is valid}\} = S_{DP}{}^D \setminus S_P{}^D$. From the clause $x \neq y$ we get that $S_P{}^D \cap S_{-P}{}^D = \emptyset$, and from the clause $f_+(x) \equiv x \vee f_-(x) \equiv x$ we get, that $S_P{}^D \cup S_{-P}{}^D = S_{DP}{}^D$.

Thus $\{d \mid -P^D(d) \text{ is valid}\} = S_{-P}{}^D$.

3.6.4 Lemma. Deleting (P,S_P) and $(-P,S_{-P})$ is sound and complete in a special case.

Rule RSC4:

IN a) SIG $P \in \mathbb{P}$, $SO(P) = S_{DP}$. For every ground term t:

$$S_{DP} \in GS(t) \Rightarrow S_P \in GS(t) \vee S_{-P} \in GS(t)$$

b) CS neither P nor -P occurs in CS. CS contains an equality literal

c) SC contains (P,S_P) and $(-P,S_{-P})$

OUT a) SIG′ P is removed from SIG.

b) CS′ $CS \cup \{\{\forall x{:}S_P, y{:}S_{-P}\ x \neq y\}\}$

c) SC′ $SC \setminus \{(P,S_P)\ (-P,S_{-P})\}$.

Proof.

IN → OUT: see the proof of the lemma above.

OUT → IN: Let (D,SIG′,R′) be an E-model for OUT. We can assume, that D is the image of $\mathbf{WST}_{gr}$ (under every $\mathbf{S}$-homomorphism). The condition for ground terms t imply that $S_{DP}{}^{D} = S_P{}^{D} \cup S_{-P}{}^{D}$. From the clause $x \neq y$ we get that $S_P{}^{D} \cap S_{-P}{}^{D} = \emptyset$. Now it is easy to construct an E-model for IN. ∎

We give an example, that the unrestricted deletion of (P,S_P) and $(-P,S_{-P})$ from SC may be faulty:

3.6.5 Example. Let the unsatisfiable clause set be:

{-P(x) Q(x x) }; { P(x) Q(x x) }; { -Q(a a) } ; {P(c) }. {-P(d) }.

A derivation of the empty clause is possible.

The clause set after the transformation is:

$\{x{:}S_P\ Q(x\ x)\}$; $\{x{:}S_{-P}\ Q(x\ x)\}$; {-Q(a a) }

If (P,S_P) and $(-P,S_{-P})$ are deleted from SC, then this clause set does not allow a derivation of the empty clause: all clauses are pure and the clause set is satisfiable. This example may also serve as an example, that the usage of the union of sorts may lead to undesired effects:

In the above clause set the information, that $S_P \cup S_{-P} = \top$ makes the clause set unsatisfiable, since then the constant a is either of sort S_P or of sort S_{-P}. But all the clauses remain pure in the sense of complementary unifiability. Hence the purity reduction rule is not correct in this case.

3.6.6 Lemma. Deleting (P,S_P) and $(-P,S_{-P})$ is sound and complete in a special case.

Rule RSC5:

IN a) SIG $P \in \mathbb{P}$, $SO(P) = S_{DP}$. For every ground term t:

$$S_{DP} \in GS(t) \Rightarrow (S_P \in GS(t) \Longleftrightarrow S_{-P} \notin GS(t)).$$

b) CS neither P nor -P occurs in CS. CS contains no equality literal

c) SC contains (P,S_P) and $(-P,S_{-P})$

OUT a) SIG′ P is removed from SIG.

b) CS′ CS

c) SC′ $SC \setminus \{(P,S_P)\ (-P,S_{-P})\}$.

Proof.

IN → OUT: trivial.

OUT → IN: Let (D,SIG′,R′) be an E-model for OUT. We can assume, that D is the image of $\mathbf{WST}_{gr}$ (under every $\mathbf{S}$-homomorphism). The condition for ground terms t imply that $S_{DP}{}^{D} = S_P{}^{D} \cup S_{-P}{}^{D}$ and that $S_P{}^{D} \cap S_{-P}{}^{D} = \emptyset$. Now it is easy to construct an E-model for IN. ∎

3.6.7 Lemma. Deleting intersection information from SC is sound and complete provided the appropriate clauses are added.

Rule RSC6.

IN a) SIG
b) CS
c) SC contains (S_1,S_2,T), where $S_1 \neq T$, $S_2 \neq T$ and $S_1 \wedge S_2 = T$

OUT a) SIG′ A new (skolem) function g is added to SIG where g has domain-sort S_1 and range-sort T and $(S,S) \in SO(g)$ for all $S \leq T$
b) CS′ $CS \cup \{(\forall x{:}S_1, y{:}S_2\,.\; x \neq y \vee g(x) \equiv x)\}$
c) SC′ $SC \setminus \{(S_1,S_2,T),(S_2,S_1,T)\}$.

Proof.

IN → OUT: Let (D,SIG,R) be an E-model for IN. We construct an E-model for OUT. We define $g^D(d) = d$ for every $d \in T^D$. For $d_1 \in S_1{}^D$ and $d_2 \in S_2{}^D$ either $d_1 \neq d_2$ or $d_1 = d_2$ and $d_1 \in T^D$. For both possibilities, the new clause is valid. Hence there exists an E-model for OUT.

OUT → IN: We show only, that $S_1{}^D \cap S_2{}^D = T^D$. Obviously $S_1{}^D \cap S_2{}^D \supseteq T^D$. Let $d \in S_1{}^D \cap S_2{}^D$. Then $d \neq d$ is false, hence $d = g(d)$ is true. But this means $d \in T^D$.

3.6.8 Lemma. Deleting intersection information from SC is sound and complete in a special case.

Rule RSC7.

IN a) SIG
b) CS $\equiv$ occurs only in unit-clauses. For every triple (S,T,S') and for every literal $s \equiv t$, which follows semantically ($\models$) from the equality clauses in CS, where $S \in GS(s)$ and $T \in GS(t)$ hold, there exists a term t_R, such that $S' \in GS(t_{S'})$ and $s \equiv t_{S'}$ follows semantically from the equality clauses in CS.
c) SC For every triple $(S_1,S_2,S_3) \in SC$: $S_1 \wedge S_2 = S_3$

OUT a) SIG′ SIG
b) CS′ CS
c) SC′ SC \ {all triples in SC}.

Proof.

IN → OUT: trivial.

OUT → IN: Let (D,SIG′,R′) be an E-model for OUT. Then we can assume, that $D = \mathbf{WST}_{gr} / \sim$, where $\sim$ is the congruence relation on terms defined by the unit-equalities of CS. We show, that $S_1{}^D \cap S_2{}^D = S_3{}^D$ for every triple $(S_1,S_2,S_3) \in SC$. Let $d \in S_1{}^D \cap S_2{}^D$. Then $d \equiv^D d$ is valid. There exist $t_1,t_2 \in \mathbf{WST}_{gr}$, such that $S_1 \in GS(t_1)$, $S_2 \in GS(t_2)$ and $t_1 \sim t_2$. The condition of RSC7 implies, that there exist a term t_3 of sort $S_3 = S_1 \wedge S_2$ and $t_3 \sim t_2$. Hence $t_3{}^D = d \in S_3{}^D$. ■

3.6.9 Example. The creation of intersection of sorts may be incomplete, if this information is not coded in clauses. We give an unsorted contradictory clause set and transform it in a sorted one, which is satisfiable, if the intersection clause is missing. The clause set is:

$A(f(x_1))$; $B(g(x_2))$; $f(a) \equiv g(b)$; $A(x_3) \wedge B(x_3) \Rightarrow -P(x_3,x_3)$; $P(f(a), f(a))$; $A(c)$; $B(c)$.

The empty clause is deducable, since $A(f(a))$ and $B(f(a))$ are deducable, and hence $-P(f(a), f(a))$ is deducable.

After the transformation, we have the sort structure $\mathbb{S} = \{\top,S_A,S_B,S_C\}$ with $S_A \geq S_C$ and $S_B \geq S_C$. The signature contains the information: $f: \top \to S_A$; $g: \top \to S_B$; $c{:}C$; $a{:}\top$; $b{:}\top$

The clauses are:

$f(a) \equiv g(b)$; $-P(x_4,x_4)$ (where $x_4{:}S_C$); $P(f(a), f(a))$.

Paramodulation into $-P(x_4,x_4)$ is not possible, since the x_4 is not unifiable with $f(a)$ or $g(b)$. Paramodulation into the third clause is possible, the first argument of all paramodulants is either f(a) or g(b).The empty clause is not deducable, since $-P(x_4,x_4)$ and P(f(a),...) respectively P(g(b),...) are not unifiable. The reason for this incompleteness is, that S_C^D is not forced to be identical with $S_A^D \cap S_B^D$ in an E-model. If we add the clause $\forall x{:}S_A, y{:}S_B\ x \neq y \vee h(x) \equiv x$, where $h{:}S_A \to S_C$ is a new function, then a deduction of the empty clause is possible, since we can deduce $h(f(a)) \equiv f(a)$ and P(h(f(a)), h(f(a))). The latter is unifiable with $-P(x_4,x_4)$.

3.7 The Rules ACi are Sound and Complete.

3.7.1 Lemma. The rules AC1, AC2 and AC3 are sound and complete.
Proof. These rules are correct, since the clauses, which are added, are tautologies and hence true in every E-model. ∎

3.7.2 Lemma. Splitting a clause into two is sound and complete.

Rule AC4.

IN a) SIG $SO(P) = S_{DP}$

b) CS CS contains a clause C, such that there exists an $x \in \mathbf{V}(C)$, with $[x] = S \le S_{DP}$.

c) SC contains (P,S_P), $(-P,S_{-P})$, (S,S_P,S_1), (S,S_{-P},S_2).

OUT a) SIG′ SIG

b) CS′ $CS \setminus \{C\} \cup \{C_1, C_2\}$, where C_i is the clause C, but the variable x is replaced by $x_i{:}\,S_i$.

c) SC′ SC

Proof.

IN → OUT. trivial, since the clauses C_i are instances of the clause C.

OUT → IN. Let (D,SIG,R) be an E-model of OUT. Let $\varphi{:}\mathbf{WST} \to D$ be an $\mathbb{S}$-homomorphism. Then $\varphi x \in S_1^D$ or $\varphi x \in S_2^D$, since either $P^D(\varphi x)$ is valid or $-P^D(\varphi x)$ is valid. We assume w.l.o.g., that $\varphi x \in S_1^D$. Then an $\mathbb{S}$-homomorphism $\varphi_1{:}\ \mathbf{WST} \to D$ exists with $\varphi_{|\mathbf{V}(C)} = \varphi_{1|\mathbf{V}(C)}$ and $\varphi x = \varphi_1 x_1$. We have $\varphi C = \varphi_1 C_1$, hence φC is valid. ∎

3.8 Sort Manipulations Caused by Equalities.

3.8.1 Lemma. Rule EQ1 is sound and complete.

Rule EQ1.

IN a) SIG

b) CS CS contains a clause $\{s \equiv t\}$, $S \in GS(s)$, $T \in GS(t)$ and $S \cap T = \emptyset$

c) SC

OUT a) SIG′ $\mathbb{S}' = \mathbb{S} \cup \{S_N\}$, S_N is a new sort symbol, c is a new constant of sort S_N. $S_N \le S$ and $S_N \le T$ is added. . $\le'$ is the transitive closure of $\le$.

b) CS′ CS

c) SC′ $SC \cup \{(S,T,S_N)\}$.

Proof.

IN → OUT. Let (D,SIG,R) be an E-model of IN . The nontrivial part is to show, that $S^D \cap T^D \neq \emptyset$. For every $\mathbb{S}$-homomorphism $\varphi{:}\ \mathbf{WST} \to D$, we have $\varphi s = \varphi t$, hence $\varphi s \in S^D \cap T^D = S_N^D$.

OUT → IN. trivial. ∎

3.8.2. Lemma. Rule EQ2 is sound and complete.

Rule EQ2.

IN a) SIG contains a constant c of sort S_c.

b) CS CS contains a clause $\{c \equiv t\}$, $S_t \in GS(t)$.

c) SC contains (S_c,S_t,S_{ct}).

OUT a) SIG′ the sort of c is changed into S_{ct}.

b) CS′ CS

c) SC′ SC

Proof.

IN → OUT. Let (D,SIG,R) be an E-model of IN We have $c^D \in S_c^D \cap S_t^D = S_{ct}^D$.

OUT → IN. trivial.■

3.8.3 Lemma. Rule EQ3 is sound and complete.

Rule EQ3.

IN a) SIG

b) CS CS contains a clause $\{x \equiv t\}$, $T \in GS(t)$. and x is a variable of sort S.

c) SC

OUT a) SIG′ $S \leq T$ is added. $\leq'$ is the transitive closure of $\leq$.

b) CS′ CS

c) SC′ SC

Proof.

IN → OUT. Let (D,SIG,R) be an E-model of IN. For every $d \in S^D$ there exists an $\mathbf{S}$-homomorphism φ: **WST** → D , such that $\varphi x = d$. We have $d = \varphi t$, hence $d \in T^D$. We conclude $S^D \subseteq T^D$.

OUT → IN. trivial.■

3.8.4 Lemma. Rule EQ4 is sound and complete.

Rule EQ4.

IN a) SIG $(S_1,\ldots,S_{n+1}) \in SO(f)$

b) CS CS contains a clause $\{f(x_1,\ldots,x_n) \equiv t\}$, $T \in GS(t)$. and the x_i are distinct variables of sort S_i.

c) SC contains (S_{n+1},T,S')

OUT a) SIG′ $SO'(f) = SO(f) \cup \{(S_1,\ldots,S_n,S')\}$.

b) CS′ CS

c) SC′ SC

Proof.

IN → OUT. Let (D,SIG,R) be an E-model of IN. For every $d_i \in S_i^D$ there exists an $\mathbf{S}$-homomorphism φ: **WST** → D , such that $\varphi x_i = d_i$. $\varphi(f(x_1,\ldots,x_n) \equiv t)$ is valid, hence $f^D(d_1,\ldots,d_n) \in S_{n+1}{}^D \cap T^D = S'^D$.

OUT → IN. trivial. ■

3.9 Decision Rules for Clause Sets with Unary Predicates and Unary Functions.

3.9.1 Lemma. Rule UC1 is sound and complete.

Rule UC1.

IN a) SIG $SO(P) = S_{DP}$.

b) CS CS contains a literal $\pm P(g(t))$

c) SC

OUT a) SIG′ P_g is a new predicate with $SO(P_g) = (\top)$.

b) CS′ $CS'' \cup \{(\forall x_i{:}S_i\ -P(g(x_i)) \vee P_g(x_i))\} \cup \{(\forall x_i{:}S_i\ P(g(x_i)) \vee -P_g(x_i))\}$ for all i, where $\{S_1,\ldots,S_n\} = MAX_{\leq}\{T_1 \mid (T_1,T_2) \in SO(g)\ ,T_2 \leq S_{DP}\}$.

CS″ is the clause set CS, where all literals of the form $\pm P(g(t))$ are replaced by the $\pm P_g(t)$.

c) SC′ SC

Proof.

IN → OUT: Let (D,SIG,R) be an E-model of IN. Note that the new clauses are well-sorted. For $d \in D$, let $P_g(d)$ be valid, iff $P^D(g^D(d))$ is valid. We have constructed an E-model of OUT.

OUT → IN. The added clauses guarantee, that $P_g(d)$ is valid, iff $P^D(g^D(d))$ is valid. The rest is trivial.■

3.9.2 Lemma. For a clause set, where all predicates and functions are unary (no equalitiy literals are allowed), there exists a sequence of applications of rules of SOGEN, such that a set $\{ (SIG_i,CS_i,SC_i), i=1,...,n \}$ (i.e. splitparts) is produced and $CS_i = \emptyset$ for all i. The initial clause set is contradictory, iff all elements of this sets are contradictory.

Proof. With rule UC1, it is possible to transform the clause set, until all clauses with an occurrence of a function are among the clauses, which are added by UC1. By case analysis (rule AC1), for every unary predicate we can introduce sorts. Obviously, all clauses are deleted and coded into the signature and SC. Now the rules (AC3 + BT), SOi,SCi, and MS have to be applied until the rule AC3 is not applicable and the signature is polymorphic. Then Lemma 3.10.1 is applicable. ■

3.10 Termination of SOGEN.

3.10.1 Lemma. Let SIG be a polymorphic signature. Let CS be the empty clause set. Then an E-model for CS,SC exists , iff for every two pairs (P,S_p), $(-P,S_{-p}) \in SC$ and all sorts $S \in \mathbb{S}$: $S \le S_P \iff \neg(S \le S_{-p})$.

Proof. "⇒": The only-if part is trivial.

"⇐": Let the conditions above be satisfied. We have to construct an E-model for SIG,CS, SC. Let $S_1,...,S_n$ be the minimal sorts of $\mathbb{S}$. Let $D = \{s_i \mid i=1,...,n\}$ where all s_i are different elements. We define the algebra (D,SIG):

$S_i^D = \{s_i\}$. If c is a constant of sort S_c, then we choose a minimal sort $S_k \le S_c$, and define $c^D = s_k$. For a function f and $(T_1,...,T_{n+1}) \in SO(f)$, such that the T_i are minimal sorts, we choose an element $d_{n+1} \in D$ with $d_{n+1} \in S_{n+1}{}^D$. For the unique elements $d_i \in S_i{}^D$ we define $f^D(d_1,...,d_n) = d_{n+1}$. With these definitions (D,SIG) is an algebra of type SIG, since SIG is polymorphic.

We have $S^D = \bigcup_{T \le S,\ S \text{ minimal}} T^D$.

This implies that all intersection restrictions are satisfied. The pairs (P,S_P), $(-P,S_{-p})$ are equivalent with $S_P{}^D \cap S_{-p}{}^D = \emptyset$ and $S_P{}^D \cup S_{-p}{}^D = S_{DP}{}^D$, where $SO(P) = S_{DP}$. But since for all sorts $S' \le S_{DP}$ we have $S' \le S_P$ or $S' \le S_{-p}$, these conditions are satisfied. ■

3.10.2 Lemma. Given arbitrary SIG and SC, the rules MS1, SC1, SC2 and SC4 can be applied only finitely many times, provided the rules MS1 and SC2 have an higher priority than SC4. The resulting $\langle \mathbb{S}', \le' \rangle$ has the following properties:

i) For all $S_1, S_2 \in \mathbb{S}$: $S_1 \sqcap S_2 \ne \emptyset$ implies, that there exists a $S_3 \in \mathbb{S}$, such that $(S_1,S_2,S_3) \in SC$ and that $S \le S_3$ for all $S \in S_1 \sqcap S_2$. i.e either $S_1 \wedge S_2$ exists and equals $S_1 \sqcap S_2$ or $S_1 \sqcap S_2 = \emptyset$.

ii) $\langle \mathbb{S}', \le' \rangle$ is cycle-free.

Proof. The Rule MS1 does not increase an intersection base $BASE_{SC}$. The same holds for the rule SC4. Now the number of possible equivalence classes (with respect to $\sim_{SC}$) is finite. Since any relation $\sim_{SC}$ or $\le_{SC}$ is immediately transformed in a relation between sorts, and $\sim$ is immediately factored out, the application of the Rule SC4 is possible only once for every combination REP_1, REP_2 of subsets of $BASE_{SC}$. The number of such combinations is finite, hence SC4 can be applied only finitely many times. But then MS1 and SC2 are trivially applicable only finitely many times. ■

The next lemma shows, that a certain combination of rules terminates and that the resulting signature is polymorphic.

3.10.3 Lemma. For any imput IN, we can apply the rules SC1,SC2,SC3,SC4, MS1, SO1,SO2,SO3 only finitely many times, provided the rules have the priority: SO2, SO3, SC1, SC2, MS1, SC3, SC4, SO1. The resulting signature is a polymorphic one and the sort structure $\langle \mathbb{S}, \leq \rangle$ is a semilattice.

Proof.

No one of the rules mentioned above increases $BASE_{SC}$. The same arguments as in 3.10.2 show, that the rules SO2, SO3, SC1, SC2, MS1, SC3, SC4 can be applied only finitely many times. We show, that after termination of these rules:

- $\langle \mathbb{S}, \leq \rangle$ is a semilattice,
- the precondition of SO1 is satisfied.
- rule SO1 makes SO(f) polymorphic for every f.

i) That $\langle \mathbb{S}, \leq \rangle$ is a semilattice is trivial, since for every $R,S \in \mathbb{S}$: if $R \sqcap S \neq \emptyset$, then $R \wedge S$ is defined and $(R,S,R \wedge S) \in SC$.

ii) $\langle \mathbb{S}, \leq \rangle$ is cycle free, since the rule MS1 is not applicable. For S_i, T_i with $S_i \sqcap T_i \neq \emptyset$, the element $S_i \wedge T_i$ is defined and unique, since otherwise either SC4 or MS1 fires. The rule SO2 does not fire, hence for two tuples $(S_1,\ldots,S_{n+1})$ and $(T_1,\ldots,T_{n+1}) \in SO(f)$: $S_i \sqcap T_i \neq \emptyset$, $i=1,\ldots,n$, we have $S_{n+1} \sqcap T_{n+1} \neq \emptyset$. The rule SO3 then yields a R_{n+1}, such that $(S_1 \wedge T_1,\ldots, S_n \wedge T_n, R_{n+1}) \in SO(f)$.

iii) Let $(S_1,\ldots,S_{n+1}) \in SO(f)$ and let $(T_1,\ldots,T_n) \leq (S_1,\ldots,S_n)$. Then there exists a T_{n+1}, such that $(T_1,\ldots,T_{n+1}) \in SO(f)$ and $T_{n+1} \leq S_{n+1}$, since S_{n+1} is the minimal element of a set M_S and T_{n+1} of M_T, and obviously $M_S \subseteq M_T$. ■

4. A Literal Reduction Rule.

The reduction rule given here is a very complex one. It does not directly reduce the search space of a problem, but is able to detect some hidden sort-information. With this reduction rule, the results in Example 5.4 are the same in i) and ii). This means, that the sort-generation becomes stronger. Unfortunately, this reduction rule can not be transformed into deductions.

4.1 Theorem. Let CS be a clause set, $f \in \mathbb{F}$ be a (fixed) function.

Let the following conditions be satisfied:

i) $C_1,\dots,C_k$ are exactly the clauses with an occurence of f.

ii) $C_i = C_{i,0} \cup C_{i,f}$, where $C_{i,0}$ is the f-free part of C_i and $C_{i,f}$ are the literals of C_i with an occurence of f.

iii) $C_{1,f} = \{P(t_1,\dots,t_m)\}$, every subterm of $C_{1,f}$ starting with f is identical with $f(x_1,\dots,x_n)$, where the x_i's are distinct variables of maximal sort $[x_1] = S_{x,i}$, i.e. $(S_{x,1},\dots,S_{x,n},[f(x_1,\dots,x_n)])$ is the greatest element of SO(f).

iv) $\mathbf{V}(C_{1,f}) \subseteq \{x_1,\dots,x_n\}$

v) For every subterm $t_j = f(s_1,\dots,s_n)$ of $C_{j,f}$, there exists a Σ-substitution λ with $DOM(\lambda) \subseteq \mathbf{V}(C_1) \setminus \{x_1,\dots,x_n\}$ and $\lambda \cdot \{x_i \leftarrow s_i \mid i=1,\dots,n\}\, C_{1,0} \subseteq C_{j,0}$.

vi) There exists a unit clause $P(r_1,\dots,r_m)$ in CS, such that $\tau\, C_{1,f}' = P(r_1,\dots,r_m)$, where $C_{1,f}'$ is constructed from $C_{1,f}$ by replacing all terms $f(x_1,\dots,x_n)$ by a (new) variable y_0 of sort S_f, the maximal range sort of f. The matcher τ should have the properties that $\tau_{|\{x_1,\dots,x_n\}}$ is a variable renaming and $DOM(\tau) \subseteq \{x_1,\dots,x_n,y_0\}$

vii) There exists a unique minimal range sort $S_{f,min}$ of f such that $[\tau y_0] \le S_{f,min}$ (τ is the substitution of vi))

Then we can replace $C_{1,0}$ by any subset of $C_{1,0}$ without loosing soundness and completeness.

Proof.

Let CS* be the clause set after removing literals from $C_{1,0}$. We show, that CS has an E-model, iff CS* has an E-model:

The one direction is trivial.

We prove, that CS* is satisfiable, provided CS is satisfiable:

Let (D,SIG,R) be an E-model for CS.

We construct an E-model of CS*:

$$\text{Let } N = \left\{ (d_1,\dots,d_n) \;\middle|\; \begin{array}{l} d_i \in S_{x,i}{}^D.\ \text{For every } \mathbf{S}\text{-homomorphism } \varphi: \mathbf{WST} \to D, \\ \text{with } \varphi x_i = d_i,\ \varphi C_{1,0} \text{ is valid in (D,SIG,R).} \end{array} \right\}$$

We define an algebra E of type SIG with E = D, but the representation is different:

Sorts: $S^E = S^D$ for all $S \in \mathbf{S}$.

Constants: $c^E = c^D$ for all $c \in \mathbb{C}$.

Functions: For $g \neq f$, let $g^D = g^E$.

For $(d_1,\dots,d_n) \in N$: We define $f^D(d_1,\dots,d_n) = f^E(d_1,\dots,d_n)$.

For $(d_1,\dots,d_n) \notin N$: Since τ (see vi) is a variable renaming on the set $\{x_1,\dots,x_n\}$, there exists an $\mathbf{S}$-homomorphism $\varphi^D{}_{d_1,\dots d_n}: \mathbf{WST} \to D$, such that $\varphi^D{}_{d_1,\dots d_n} \cdot \tau\, x_i = d_i$

We define $f^E(d_1,\dots d_n) = \varphi^D{}_{d_1,\dots d_n}(\tau y_0)$.

The condition vii) now guarantees, that E is an algebra of type SIG, since the mapping properties of f^E are satisfied.

The set of relations R is not changed.

There is a 1-1 correspondence between $\mathbb{S}$-homomorphisms w.r.t. E and $\mathbb{S}$-homomorphisms w.r.t D. ψ^D corresponds to ψ^E, iff they are equal on all variables. Obviously $\psi^E(L) = \psi^D(L)$ for every literal, which does not contain the function symbol "f".

Now we show, that (E,SIG,R) is an E-model of CS*:
Let φ^E: $\mathbf{WST} \to E$ be an $\mathbb{S}$-homomorphism and let $\varphi^E x_i = d_i$.

a) The changes do not affect the clauses in CS \ $\{C_1,...,C_m\}$, hence they are true under all $\mathbb{S}$-homomorphisms φ^E: $\mathbf{WST} \to E$.

b) The clause $C_1{}^*$ is true in (E,SIG,R):

<u>CASE</u> $(d_1,...,d_n) \notin N$. Then an $\mathbb{S}$-homomorphism θ^D: $\mathbf{WST} \to D$ exists, such that $\theta^D x_i = \varphi^E x_i$. Hence by construction of N, there exists an $\mathbb{S}$-homomorphism ψ^D: $\mathbf{WST} \to D$, such that $\psi^D(C_{1,0})$ is not valid. But then $\psi^D C_{1,f}$ is true, because (D,SIG,R) is an E-model. From the conditions iii) iv) and the definition of f^E it follows, that $\psi^D C_{1,f} = \varphi^E C_{1,f}$, hence $\varphi^E C_{1,f}$ is valid in (E,SIG,R).

<u>CASE</u> $(d_1,...,d_n) \in N$. We denote by $\varphi_d{}^E$: $\mathbf{WST} \to E$ the $\mathbb{S}$-homomorphism, which is identical with $\varphi^D{}_{d1,...dn}$: $\mathbf{WST} \to D$ on all variables.

We have $\varphi^E(x_i) = \varphi_d{}^E \circ \tau(x_i) = d_i$ and

$\varphi^E f(x_1,...x_n) =$

$= f^E(d_1,...d_n)$ (definition of an $\mathbb{S}$-homomorphism)

$= \varphi^D{}_{d1,...dn}(\tau y_0)$ (definition of f^E and $(d_1,...d_n) \in N$)

$= \varphi^E{}_d(\tau y_0)$ (τy_0 is f-free)

Hence we have:

$\varphi^E C_{1,f} =$

$= \varphi^E{}_d \circ \tau\ (C_{1,f}{}')$ ($\varphi^E(f(x_1,...x_n)) = \varphi^E{}_d \circ \tau(y_0)$ and $\varphi^E(x_i) = \varphi_d{}^E \circ \tau\ (x_i)$)

$= \varphi^E{}_d(P(r_1,...,r_m))$ (see condition vi))

$= \varphi^D{}_{d1,...dn}\ (P(r_1,...,r_m))$ ($P(r_1,...,r_m)$ does not contain f).

This shows, that $C_{1,f}$ is valid in (E,SIG,R) under φ^E.

iii) C_j is true in (E,SIG,R) for $j \geq 2$:

<u>CASE</u> For every subterm $f(s_1,...s_n)$ of $C_{j,f}$ we have $(\varphi^E s_1,..., \varphi^E s_n) \notin N$.:

Then for the corresponding $\mathbb{S}$-homomorphism φ^D: $\mathbf{WST} \to D$ ($\varphi^D x = \varphi^E x$ for all variables x) we have $\varphi^D C_j = \varphi^E C_j$, since $f^E\ (\varphi^E s_1,..., \varphi^E s_n) = f^D\ (\varphi^E s_1,..., \varphi^E s_n)$. Hence C_j is true in (E,SIG,R).

<u>CASE</u> A subterm $f(s_1,...s_n)$ of $C_{j,f}$ exists such that $(\varphi^E s_1,..., \varphi^E s_n) \in N$.

Let $\sigma = \{x_1 \leftarrow s_i, i=1,...n\}$. σ is an $\mathbb{S}$-substitution. From condition v), we have that there exists an $\mathbb{S}$-substitution λ with $\lambda \circ \sigma\ C_{1,0} \subseteq C_{j,0}$.

The following equalities hold:

$\varphi^E\ \lambda \circ \sigma\ x_i =$

$= \varphi^E\ \lambda s_i$ ($\sigma\ x_i = s_i$)

$= \varphi^E\ s_i$ ($DOM(\lambda) \cap \mathbf{V}(s_i) = \emptyset$).

Now $\varphi^E(\lambda \circ \sigma\ C_{1,0})$ is valid, since $(\varphi^E\ \lambda \circ \sigma\ x_1,..., \varphi^E\ \lambda \circ \sigma\ x_n) \in N$,$\varphi^E\ \lambda \circ \sigma$ is an $\mathbb{S}$-homomorphism (= $\varphi^D\ \lambda \circ \sigma$ on $C_{1,0}$) and $C_{1,0}$ is f-free. Hence $\varphi^E\ (C_{j,0})$ is also true. ∎

5. Examples

In this section some examples are given, which demonstrate the power of SOGEN:

5.1 Schubert's Steamroller [Wa84]

The problem of Schubert reads as follows:
Wolves, foxes, birds, caterpillars, and snails are animals. Grains are plants. There exist wolves, foxes, birds, caterpillars, snails, and grains.
Every animal eats all plants or any smaller animals that eat some plants.
Birds are smaller than foxes which in turn are smaller than wolves. Wolves do not eat foxes or grains. Birds eat caterpillars, but no snails. Caterpillars and snails eat some plants.
The theorem to prove is:
There is a grain eating animal that is eaten by another animal.

Here is a axiomatization in first order predicate logic (without sorts):

WOLF (x) ⇒ ANIMAL (x) ;
FOX (x) ⇒ ANIMAL (x) ;
BIRD (x) ⇒ ANIMAL (x) ;
CATERPILLAR (x) ⇒ ANIMAL (x) ;
SNAIL (x) ⇒ ANIMAL (x) ;
GRAIN (x) ⇒ PLANT (x) ;
WOLF (LUPO) ∧ FOX (FOXY) ∧ BIRD (TWEEDY) ∧ CATERPILLAR (MAGGIE)
∧ SNAIL (SLIMEY) ∧ GRAIN (STALKY) ;

∀w: ANIMAL(w) ⇒ ((∀x PLANT (x) ⇒ EATS (w x)) ∨
((∀y: ANIMAL(y) ∧ SMALLER (y w) ∧ (∃z: PLANT (z) ∧ EATS(y z)))
⇒ EATS (w y)) :

CATERPILLAR (x) ∧ BIRD (y) ⇒ SMALLER (x y);
SNAIL (x) ∧ BIRD (y) ⇒ SMALLER (x y);
BIRD (x) ∧ FOX (y) ⇒ SMALLER (x y);
FOX (x) ∧ WOLF (y) ⇒ SMALLER (x y) ;
WOLF (x) ∧ FOX (y) ⇒ ¬ EATS(x y);
WOLF (x) ∧ GRAIN (y) ⇒ ¬ EATS(x y);
BIRD (x) ∧ CATERPILLAR (y) ⇒ EATS(x y);
BIRD (x) ∧ SNAIL (y) ⇒ ¬ EATS(x y);

CATERPILLAR (x) ⇒ (∃y: PLANT (y) ∧ EATS (x y));
SNAIL (x) ⇒ (∃y: PLANT (y) ∧ EATS (x y));

¬ EATS (x x);
ANIMAL (x) ⟺ ¬ PLANT (x) ;

Theorem:
∃x,y: ANIMAL (x) ∧ ANIMAL (y) ∧ EATS (x y) ∧ (∀z GRAIN (z) ⇒ EATS (y z)

Normalization and skolemization yields the clauses:
Ax1 -WOLF (x), ANIMAL (x) ;
Ax2 -FOX (x), ANIMAL (x) ;
Ax3 -BIRD (x), ANIMAL (x) ;
Ax4 -CATERPILLAR (x), ANIMAL (x) ;
Ax5 -SNAIL (x), ANIMAL (x) ;
Ax6 -GRAIN (x), PLANT (x) ;
Ax7 WOLF(LUPO) ;
Ax8 FOX(FOXY) ;
Ax9 BIRD (TWEEDY);
Ax10 CATERPILLAR (MAGGIE) ;

Ax11 SNAIL (SLIMEY) ;
Ax12 GRAIN (STALKY) ;
Ax13 -ANIMAL(w), -PLANT(x), EATS(w x), -ANIMAL(y), -SMALLER(y w),
-PLANT (z), -EATS(y z), EATS(w y) ;
Ax14 -CATERPILLAR (x), -BIRD (y), SMALLER(x y) ;
Ax15 -SNAIL(x), -BIRD(y), SMALLER (x y) ;
Ax16 -BIRD (x), -FOX(y), SMALLER(x y) ;
Ax17 -FOX(x), -WOLF(y), SMALLER(x y) ;
Ax18 -WOLF(x), -FOX(y), -EATS(x y) ;
Ax19 -WOLF(x), -GRAIN(y), -EATS(x y) ;
Ax20 -BIRD(x), -CATERPILLAR(y), EATS(x y) ;
Ax21 -BIRD(x), -SNAIL(y), -EATS(x y) ;
Ax22 -CATERPILLAR(x), PLANT(f_1(x)) ;
Ax23 -CATERPILLAR(x), EATS(x f_1(x)) ;
Ax24 -SNAIL(x), PLANT(f_2(x)) ;
Ax25 -SNAIL(x), EATS(x f_2(x)) ;
Ax26 ANIMAL(x), PLANT(x) ;
Ax27 -ANIMAL(x), -PLANT(x) ;
Ax28 -EATS (x x) ;

Th1 -ANIMAL(x), -ANIMAL(y), -EATS(x y), GRAIN(f_3(y x)) ;
Th2 -ANIMAL(x), -ANIMAL(y), -EATS(x y), -EATS(y f_3(y x)) ;

The Automated Theorem Prover(ATP) MKRP [KM84] has found a contradiction after 55 resolution steps. This proof uses only unit-resolution steps and was actually found by the Terminator-module [AO83].

This clause set was transformed by SOGEN into it's sorted version. The resulting signature and clauses are:

Sorts: $\top \geq$ S+ANIMAL, S+PLANT,
S+ANIMAL $\geq$ S+WOLF, S+FOX, S+BIRD, S+CATERPILLAR, S+SNAIL
S+PLANT $\geq$ S+GRAIN

Constants: LUPO: S+WOLF; FOXY: S+FOX; TWEEDY: S+BIRD;
MAGGIE: S+CATERPILLAR; SLIMEY: S+SNAIL; STALKY: S+GRAIN.

Functions: f_1: S+CATERPILLAR $\rightarrow$ S+PLANT
f_2: S+SNAIL $\rightarrow$ S+PLANT
f_3: S+ANIMAL $\times$ S+ANIMAL $\rightarrow$ S+GRAIN.

Clauses:
IC1 (Ax28) x:$\top$ -EATS(x x)
IC2 (Ax23) x:S+CATERPILLAR +EATS(x f_1(x))
IC3 (Ax25) x:S+SNAIL +EATS(x f_2(x))
IC4 (Ax14 x:S+CATERPILLAR, y:S+BIRD +SMALLER(x y)
IC5 (Ax15) x:S+SNAIL, y:S+BIRD +SMALLER(x y)
IC6 (Ax16) x:S+BIRD, y:S+FOX +SMALLER(x y)
IC7 (Ax17) x:S+FOX, S+WOLF +SMALLER(x y)
IC8 (Ax18) x:S+WOLF, x:S+FOX -EATS(x y)
IC9 (Ax19) x:S+WOLF, x:S+GRAIN -EATS(x y)
IC10 (Ax20) x:S+BIRD, x:S+CATERPILLAR +EATS(x y)
IC11 (Ax21) x:S+BIRD, x:S+SNAIL -EATS(x y)
IC12 (Ax13) x,y:S+ANIMAL z,u:S+PLANT +EATS(x u) -SMALLER(y x)
-EATS(y z) +EATS(x y)
IC13 (Th2) x,y:S+ANIMAL -EATS(y x) -EATS(x f_3(x y))

The MKRP Theorem Prover found a (unit-) refutation for this clauses set after 11 steps (including 10 resolutions and one factorization). The used CPU-time for the transformation and the search for the proof in the sorted clause set was remarkably shorter than the search for the proof in the unsorted version.

We note some difficulties in getting this result from SOGEN.

1) The theorem clause Th1 was deleted by the literal reduction rule mentioned in chapter 4.
2) To obtain the signature of the functions f_1, f_2, and f_3 the restriction rule is neccessary (SO4).
3) The rule SC3 is needed to identify the sorts S-PLANT ,S+ANIMAL and S-ANIMAL, S+PLANT.
4) The transformation is complete , since the preconditions of rule RSC4 are satisfied. ■

5.2 The Lion & Unicorn Examples

These examples are taken from "What is the Name of This Book" [SM78], which appears to be a goldmine for theorem proving examples. During a course on automated theorem proving in the last semester, our students had to translate these puzzles into first order predicate logic and to solve them with our theorem prover (Markgraf Karl Refutation Procedure) [KM84]. Two of these problems (Problem 47 + 48) read as follows: "When Alice entered the forest of furgetfulness, she did not forget everything, only certain things. She often forgot her name, and the most likely to forget was the day of the week. Now, the lion and the unicorn were frequent visitors to this forest. These two are strange creatures. The lion lies on Mondays, Tuesdays and Wednesdays and tells the truth on the other days of the week. The unicorn, on the other hand lies on Thursdays, Fridays and Saturdays, but tells the truth on the other days of the week."

Problem 47: One day Alice met the lion and the unicorn resting under a tree. They made the following statements:

Lion: Yesterday was one of my lying days.

Unicorn: Yesterday was one of my lying days.

From these statements, Alice who was a bright girl, was able to deduce the day of the week. What was it?

Problem 48: On another occasion Alice met the Lion alone. He made the following two statements:

1) I lied yesterday
2) I will lie again tomorrow.

What day of the week was it?

We use the predicates MO(x), TU(x), ... , SO(x) for saying that x is a Monday, Tuesday etc. Furthermore we need the binary predicate MEMB, indicating set Membership and a 3-ary predicate LA. LA(x y z) is true if x says at day y that he lies at day z, LDAYS(x) denotes the set of lying days of x. The remaining symbols are self explaining. One-character symbols like u,x,y,z are regarded as universally quantified variables.

Axiomatization of the days of the week:

MO(x)	⇔	¬(TU(x) ∨ WE(x) ∨ TH(x) ∨ FR(x) ∨ SA(x) ∨ SU(x))
TU(x)	⇔	¬(WE(x) ∨ TH(x) ∨ FR(x) ∨ SA(x) ∨ SU(x) ∨ MO(x))
WE(x)	⇔	¬(TH(x) ∨ FR(x) ∨ SA(x) ∨ SU(x) ∨ MO(x) ∨ TU(x))
TH(x)	⇔	¬(FR(x) ∨ SA(x) ∨ SU(x) ∨ MO(x) ∨ TU(x) ∨ WE(x))
FR(x)	⇔	¬(SA(x) ∨ SU(x) ∨ MO(x) ∨ TU(x) ∨ WE(x) ∨ TH(x))
SA(x)	⇔	¬(SU(x) ∨ MO(x) ∨ TU(x) ∨ WE(x) ∨ TH(x) ∨ FR(x))
SU(x)	⇔	¬(MO(x) ∨ TU(x) ∨ WE(x) ∨ TH(x) ∨ FR(x) ∨ SA(x))

Axiomatization of the function yesterday:

MO(yesterday(x))	⇔	TU(x)
TU(yesterday(x))	⇔	WE(x)
WE(yesterday(x))	⇔	TH(x)
TH(yesterday(x))	⇔	FR(x)

$FR(yesterday(x)) \Leftrightarrow SA(x)$
$SA(yesterday(x)) \Leftrightarrow SU(x)$
$SU(yesterday(x)) \Leftrightarrow MO(x)$

Axiomatization of the function two-after:

$MO(two\text{-}after(x)) \Leftrightarrow FR(x)$
$TU(two\text{-}after(x)) \Leftrightarrow SA(x)$
$WE(two\text{-}after(x)) \Leftrightarrow SU(x)$
$TH(two\text{-}after(x)) \Leftrightarrow MO(x)$
$FR(two\text{-}after(x)) \Leftrightarrow TU(x)$
$SA(two\text{-}after(x)) \Leftrightarrow WE(x)$
$SU(two\text{-}after(x)) \Leftrightarrow TH(x)$

Axiomatization of the function LDAYS:

$MEMB(x\ LDAYS(lion)) \Leftrightarrow MO(x) \vee TU(x) \vee WE(x)$
$MEMB(x\ LDAYS(unicorn)) \Leftrightarrow TH(x) \vee FR(x) \vee SA(x)$

Axiomatization of the predicate LA:

$\neg MEMB(x\ LDAYS(u)) \wedge LA(u\ x\ y) \Rightarrow MEMB(y\ LDAYS(u))$
$\neg MEMB(x\ LDAYS(u)) \wedge \neg LA(u\ x\ y) \Rightarrow \neg MEMB(y\ LDAYS(u))$
$MEMB(x\ LDAYS(u)) \wedge LA(u\ x\ y) \Rightarrow \neg MEMB(y\ LDAYS(u))$
$MEMB(x\ LDAYS(u)) \wedge \neg LA(u\ x\ y) \Rightarrow MEMB(y\ LDAYS(u))$

Theorem of Problem 47:

$\exists x\ LA(lion\ x\ yesterday(x)) \wedge LA(unicorn\ x\ yesterday(x))$

Theorem of Problem 48:

$\exists x\ LA(lion\ x\ yesterday(x)) \wedge LA(lion\ x\ two\text{-}after(x))$

The MKRP proof procedure at Kaiserslautern found a proof for the unsorted version of problem 47 after 183 resolution steps, among them 81 unnecessary steps, hence the final proof was 102 steps long. This proof contains a lot of trivial steps corresponding to common sense reasoning (like: if today is Monday, it is not Tuesday etc.).

Later the sort structure and the signature of the problem 47 was generated automatically by SOGEN.

The sort structure and the signature contain all the relevant information about the relationship of unary predicates (like our days) and the domain-rangesort relation of functions. The sort structure of the subsorts of DAYS in our example is equivalent to the lattice of subsets of {Mo, Tu, We, Th, Fr, Sa, Su} without the empty set, ordered by the subset order. Hence there are 127 ($=2^7-1$) sorts. The functions "yesterday" and "two-after" are polymorphic functions with 127 domain-sort relations. For example: yesterday ({MO, WE}) = {SU, TU}.

The unification algorithm exploits this information and produces only unifiers, which respect the sort relations, i.e. $\{x \leftarrow t\}$ is syntactically correct, if and only if the sort of the term t is less or equal the sort of the variable x. We give an example for unification: the unifier of $x{:}SO{+}TU$ and $yesterday(y{:}MO{+}TU)$ is $\{x \leftarrow yesterday(y_1{:}\ MO)\ ;\ y \leftarrow y_1{:}MO\ \}$.

The MKRP theorem-proving system [KM84] has proved the theorem of both problems in the sorted version immediately without any unnecessary steps. The length of the proof of problem 47 is 6, whereas the length of the proof of problem 48 is 4. As the protocol shows, the final substitution into the theorem clause (Problem 48) was $\{x \leftarrow y{:}MO\}$. Thus the ATP has found the answer, Monday, in a very straight forward and humanlike way. A proof protocol for problem 47 can be found in [Sch85]. We give a proof protocol for Problem 48:

C1 All x:Mo MEMB (x LDAYS(lion))
C2 All x:Tu MEMB (x LDAYS(lion))
C3 All x:We MEMB (x LDAYS(lion))

C4	All x,y:Days z:Animal	MEMB(y LDAYS(z)) MEMB (x LDAYS(z)) -LA(z y x)
C5	All x,y:Days z:Animal	MEMB(y LDAYS(z)) -MEMB(x LDAYS(z)) LA(z y x)
C6	All x,y:Days z:Animal	-MEMB(y LDAYS(z)) MEMB(x LDAYS(z)) LA(z y x)
C7	All x,y:Days z:Animal	-MEMB(y LDAYS(z)) -MEMB(x LDAYS(z)) -LA(z y x)
C8	All x:Th+Fr+Sa+Su	-MEMB(x LDAYS(lion))
Th1	All x:Days	-LA(lion x yesterday(x)) -LA(lion x two-after(x))

Proof:

C1,1 & C6,1 → R1: All x:Mo y:Th+Fr+Sa+Su MEMB(y LDAYS(lion)) LA(lion x y)
R1,2 & C8,1 → R2: All x:Mo y:Th+Fr+Sa+Su LA(lion x y)
R2,1 & Th1,2 → R3: All x:Mo -LA(lion x yesterday(x))
R3,1 & R2,1 → R6: □

6. Extension of SOGEN to Well-Formed Formulas.

In this chapter some special rules for introducing sorts in wff's are given. The mixed application of sort-generation, simplification, normalization and skolemization has the advantage, that the generated clause set is simpler and that more unary predicates can be transferred into sorts. We introduce the rules in an informal way. We give no rules for simplification, normalization or skolemization. All proofs , that these rules are sound and complete, are omitted, since we are sure, that these proofs are straight forward.

Remark. A polymorphic signature is the basis for the logic. wff's are formed in the usual way with the junctors $\neg,\wedge,\vee,\Rightarrow,\Leftrightarrow$ and the quantifiers $\forall,\exists$, where all terms and literals are well-sorted. TRUE, FALSE are nullary predicates, which denote the corresponding truth-values.

Remark. We assume, that the wff W is the input into a Theorem Prover, which tests W for satisfiability or unsatisfiability. If $W = W_1 \wedge \ldots \wedge W_n$, and some W_i is a clause, then the rules of SOGEN can be applied to W_i.

6.1 Rules for Wff's:

We use the set SC with the same meaning as in SOGEN.

i) If $(P,S_P) \in SC$ and $(S_P,S_x,S_0) \in SC$, then
$(\forall x{:}S_x\ {-P(x)} \vee A) \quad \rightarrow \quad (\forall x{:}S_0\ \mathrm{FALSE} \vee A)$

ii) If $(P,S_P) \in SC$ and $(S_P,S_x,S_0) \in SC$, then
$(\exists x{:}S_x\ {-P(x)} \wedge A) \quad \rightarrow \quad (\exists x{:}S_0\ \mathrm{TRUE} \wedge A)$

iii) If $(P,S_P) \in SC$ and $[t] \le S_P$, then
$P(t) \quad \rightarrow \quad \mathrm{TRUE}$

iv) If $(P,S_P) \in SC$ and $[t] \le S_P$, then
$-P(t) \quad \rightarrow \quad \mathrm{FALSE}$

v) If $(P,S_P) \in SC$ and $S_0 \ge S_P$, then
$(\forall x{:}S_0\ {-P(x)} \wedge A) \quad \rightarrow \quad \mathrm{FALSE}$

vi) If $(P,S_P) \in SC$ and $S_0 \ge S_P$, then
$(\exists x{:}S_0\ P(x) \vee A) \quad \rightarrow \quad \mathrm{TRUE}$

vii) $(\forall x{:}S\ A \wedge B) \quad \rightarrow \quad (\forall x{:}S\ A) \wedge (\forall x{:}S\ B)$

viii) $(\exists x{:}S\ A \vee B) \quad \rightarrow \quad (\exists x{:}S\ A) \vee (\exists x{:}S\ B)$

6.2 Example. " Andrew's Little" [EW83].

The formula W is :
$\{ (\forall x_1\ Q(x_1)) \Leftrightarrow (\exists x_2\ Q(x_2)) \} \Leftrightarrow \{ \exists x_3\ (\forall x_4\ Q(x_3) \Leftrightarrow Q(x_4)) \}$

1) We use Rule AC1 for Q, that means:
$((-Q,S_{-Q}) \in SC$ and $S_{-Q} = \top)$ or $(Q,S_Q) \in SC$

CASE 1. $(-Q,S_{-Q}) \in SC$ and $S_{-Q} = \top)$

Then $W = \{\mathrm{FALSE} \Leftrightarrow \mathrm{FALSE}\} \Leftrightarrow \{ \exists x_3\ (\forall x_4\ \mathrm{FALSE} \Leftrightarrow \mathrm{FALSE}) \}$,
which evaluates to true.

CASE 2. $(Q,S_Q) \in SC$.

Then $W = \{ (\forall x_1\ Q(x_1)) \Leftrightarrow \mathrm{TRUE} \} \Leftrightarrow \{ \exists x_3\ (\forall x_4\ Q(x_3) \Leftrightarrow Q(x_4)) \}$

CASE 2.1 $S_Q = \top$.

Then $W = \mathrm{TRUE} \Leftrightarrow \{ \exists x_3\ (\forall x_4\ \mathrm{TRUE} \Leftrightarrow \mathrm{TRUE}) \}$, which evaluates to true.

CASE 2.2 $(-Q,S_{-Q}) \in SC$. Then:

$$\text{FALSE} \Leftrightarrow \{\exists x_3\, (\forall x_4\, Q(x_3) \Leftrightarrow Q(x_4))\} \longrightarrow$$
$$-\{\exists x_3\, (\forall x_4\, Q(x_3) \Leftrightarrow Q(x_4))\} \longrightarrow$$
$$-\{\exists x_3\, (\forall x_4\, (-Q(x_3) \vee Q(x_4)) \wedge (Q(x_3) \vee -Q(x_4)))\} \longrightarrow$$
$$-\{\exists x_3\, (\forall x_4\, (-Q(x_3) \vee Q(x_4))) \wedge (\forall x_5\, (Q(x_3) \vee -Q(x_5)))\} \longrightarrow$$
$$-\{\exists x_3\, (\forall x_4{:}S_{-Q}\ -Q(x_3)) \wedge (\forall x_5{:}S_Q\, (Q(x_3)))\} \longrightarrow$$
$$-\{\exists x_3\ -Q(x_3) \wedge Q(x_3)\} \longrightarrow$$
$$\text{FALSE}.$$

6.3 Example. We demonstrate, how a formula, which occurs in the first order formulation of "Schuberts Steamroller" [Wa83], is normalized and skolemized using different methods:

We have the clauses $G(G_0)$; $A(A_0)$ and

$\forall x,y\ -A(x) \vee -E(x,y) \vee (\exists z\, G(z) \wedge -E(y,z))$

i) Sort generation after normalization.

We obtain the following clauses after normalization:

$G(G_0)$;

$A(A_0)$;

$\forall x,y\ -A(x) \vee -E(x,y) \vee G(f(x,y))$;

$\forall x,y\ -A(x) \vee -E(x,y) \vee -E(y, f(x,y))$;

Sort generation yields:

$(S_A,A) \in SC$, $A_0{:}\ S_A$;; $S_A \le \top$; $S_G \le \top$;

The clauses are:

$G(G_0)$;

$\forall x{:}S_A,\ y{:}\top\ \ -E(x,y) \vee G(f(x,y))$;

$\forall x{:}S_A,\ y{:}\top\ \ -E(x,y) \vee -E(y, f(x,y))$;

ii) Sort generation during normalization. We get:

$(S_G,G) \in SC$, $(S_A,A) \in SC$, $A_0{:}\ S_A$; $G_0{:}\ S_G$; $S_A \le \top$; $S_G \le \top$; and the clause

$\forall x{:}S_A,\ y{:}\top\ \ -E(x,y) \vee (\exists z{:}S_G\ -E(y, z))$

Skolemization then yields a function $f{:}\ (S_A,\top) \rightarrow S_G$ and the clause

$\forall x{:}S_A,\ y{:}\top\ \ -E(x,y) \vee -E(y, f(x, y))$

The difference between the two methods is that in i) the clause $\forall x{:}S_A,\ y{:}\top\ \ -E(x,y) \vee G(f(x,y))$ does contain the literal $-E(x,y)$, whereas in ii) this literal is avoided. In chapter4, we gave a reduction rule, which allows to delete such (superfluous) literals.

7. Summary.

The main results of this paper are:

i) An algorithm is described, which transforms unsorted clause sets (respectively wffs) into a sorted version. Furthermore a proof is given, that this algorithm preserves (un)satisfiability.

ii) Conditions are given for the completeness of the naive transformation (i.e. the transformation which doesn't care of intersections and complementary sorts S_P and S_{-P}).

It is not possible to give a sufficient and necessary condition for a clause set to be transformable into a sorted version. The reason is, that deduction may be necessary for such a transformation(the algorithm SOGEN makes in fact such deductions).
The algorithm SOGEN is implemented at Kaiserslautern as a preprocessor for the MKRP Automated Theorem Prover [KM84]. It has shown remarkable improvements searching for a proof in a lot of example runs.
Since this algorithm is some sense deterministic (no search) the cpu-time consumed by SOGEN is neglectable in most of the examples, but serious problems arise in cases, where the number of sorts exceeds 150. The sort structure constructed in example 5.2 is isomorphic to the lattice of subsets of a set with 7 elements (i.e. 127 sorts). I am sure that a modified implementation of sorts (computing sorts and their relations if needed) allows to handle far bigger sort structures of this type.
In the case that SOGEN fails, the cpu-time consumed by it is not totally wasted, since some of the toplevel reductions (tautologies and replacement resolution) do not depend on the success of SOGEN.

Acknowledgements.
I would like to thank: My colleagues H.J.Ohlbach, A.Herold, N.Eisinger and H.J.Buerckert for their support and their helpful discussions during the preparation of this paper; Prof. J.Siekmann, who read carefully a draft of this paper.

References.

AO83 Antoniou, G., Ohlbach, H. J.,
Terminator, Proc. of the 8th IJCAI, Karlsruhe, (1983)

CD83 Cunningham, R.J., Dick, A.J.J
Rewrite Systems on a Lattice of Types. Rep. No. DOC 83/7, Imperial College, London SW7 (1983)

Co83 Cohn, A.G.
Improving the Expressiveness of Many-sorted Logic AAAI-83, Washington (1983).

EW83 Eisinger N., Weigele M.
A Technical Note on Splitting and Clausel Form Algorithms. Proc. GWAI-83, Springer Fachberichte, 1983

GM84 Goguen, J.A. Meseguer, J.
Equalities, Types, Modules and Generics for Logic Programming, Journal of Logic Programming (1984).

GM85 Goguen, J.A. Meseguer, J.
Order Sorted Algebra I. Parial and Overloaded Operators, Error and Inheritance. SRI Report (1985).

Gr79 Grätzer, G. Universal Algebra, Springer Verlag, (1979)

Hay71 Hayes, P.
A Logic of Actions. Machine Intelligence 6, Metamathematics Unit, University of Edinburgh. (1971)

Hen72 Henschen, L.J.
N-sorted Logic for Automated Theorem Proving in Higher-Order Logic. Proc. ACM Conference, Boston (1972)

HO80 Huet, G.,Oppen, D.C.
Equations and Rewrite Rules, SRI Technical Report CSL-111, (1980)

KM84 Karl Mark G Raph,
The Markgraf Karl Refutation Procedure, Memo-SEKI-MK-84-01 (1984)

Ob62 Oberschelp, A.
Untersuchungen zur mehrsortigen Quantorenlogik. Mathematische Annalen 145 (1962)

Ro65 Robinson, J.A.
A Machine-Oriented Logic Based on the Resolution Principle. JACM 12 (1965)

Sch85 Schmidt-Schauss. M.
A Many-Sorted Calculus with Polymorphic Functions Based on Resolution and Paramodulation. Interner Bericht. Institut für Informatik, Universität Kaiserslautern (forthcoming)

Si84 Siekmann, J.H., Universal Unification, 7th Int. CADE, Napa, California (1984).

Sm78 Smullyan,
What is the Name of this Book? , Prentice Hall (1978)

Wa83 Walther C.
A Many-Sorted Calculus Based on Resolution and Paramodulation. Proc. of the 8th IJCAI, Karlsruhe, (1983)

Wa84 Walther, C.
Unification in Many-Sorted Theories. Proc. of the 6th ECAI, Pisa, (1984)

WR73 Wos, L.,Robinson, G.
Maximal Models and Refutation Completeness: Semidecision Procedures in Automatic Theorem Proving. In "Wordproblems" (W.W.Boone, F.B. Cannonito, R.C.Lyndon eds.) North-Holland (1973)